Universidad
Carlos III de Madrid
www.uc3m.es

Escuela politécnica superior

Grado en Ingeniería de Sistemas Audiovisuales

Videoconferencia con HTML5

Trabajo Fin de Grado

Autor: Ángel Muñoz Herencias

Tutor: Andrés Marín López

24/09/2014

Título: Videoconferencia con HTML5
Autor: Ángel Muñoz Herencias
Tutor: Andrés Marín López

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar me gustaría dar las gracias a mi familia por todo el apoyo que me han dado, no sólo durante el desarrollo de este fin de grado, sino a lo largo de toda la carrera y toda mi vida. En especial a mi madre Mª del Carmen por haber hecho posible que recibiera una educación, a pesar de todos los baches por los que ha tenido que pasar. También a mi hermana Sonia, por el apoyo extra que me ha transmitido durante la realización de este trabajo de fin de grado.

En segundo lugar, a mis compañeros de trabajo, en especial a Álvaro por su continuo apoyo en las últimas fases de desarrollo de este trabajo. Tampoco me olvido del equipo de "Javastrippers", ya que ellos me han transmitido todos los conocimientos sobre desarrollo web.

También quiero hacer especial mención a los compañeros que he conocido durante el transcurso de la carrera, por amenizar esta etapa de formación universitaria aportando buenos y malos momentos.

Por último, a mis amigos del pueblo, "los de toda la vida", por el apoyo demostrado y los todos los momentos vividos y los que nos quedan por vivir. También quiero dar las gracias a Marta por haber sido un apoyo fundamental durante la redacción de este documento.

Resumen

En los últimos años, el mundo de las aplicaciones web ha experimentado un gran desarrollo. Gracias a la aparición de nuevos dispositivos con acceso a internet, la consulta de contenidos web, se ha convertido prácticamente en un hábito. Esto ha hecho que cambien los hábitos de comunicación entre las personas.

Este trabajo de fin de grado tiene como objetivo el diseño y la implementación de una aplicación web que permita establecer una videoconferencia entre distintos usuarios de manera que pueda ser utilizada en cualquier dispositivo con un navegador con conexión a Internet. Para ellos se hará uso de la tecnología HTML5.

Palabras clave:

Videoconferencia, aplicaciones web, sistemas de comunicación a tiempo real, herramientas de desarrollo web.

Abstract

In the recent years, the world of web applications has experienced great development. Thanks to the emergence of new devices with internet access, web content consulting has almost become a habit, which has a result changed communication habits between people.

This dissertation has the aim of designing and implementing a web application that allows video conferencing between different users so it can be used in any device with a browser and internet connection. To carry out this aim, HTML5 technology will be used.

Keywords:

Web conference, web applications, real-time communication systems, web development frameworks.

Índice general

Índice de ilustraciones

Índice de tablas

Capítulo 1: Introducción y objetivos

1.1 Introducción

Durante los últimos años, el sector de las tecnologías de la información ha experimentado un crecimiento exponencial, y en particular el mundo de la World Wide Web. Desde los inicios de internet, la web ha ido evolucionado pasando por diferentes fases, Web 1.0, 2.0 y 3.0, términos que se han dado de acuerdo a su evolución, marcando una diferencia entre el antes y el después.

WEB 1.0	WEB 2.0	WEB 3.0
Personas conectadas a la Web	Personas conectándose a personas	Aplicaciones Web conectándose a aplicaciones Web
	Redes sociales Wikis Colaboración Posibilidad de compartir	Web Geoespacial Web Semántica Web Multimedia
	Necesidad de un gran espacio de tiempo y trabajo en las búsquedas	Búsquedas más precisas e inteligentes
	Información sin significado	Información con significado

Fuente: Andrés Richero. http://sadymaureria.wordpress.com/category/web-semantica/

Tabla 1: Evolución y características de la web

Partiendo de esta base, el trabajo de fin de grado que nos ocupa trata sobre el desarrollo de una aplicación de la denominada web 3.0. Esta aplicación será capaz de establecer una videoconferencia entre dos usuarios.

Además del desarrollo de la aplicación de videoconferencia en sí, este proyecto se centra en el uso de las nuevas tecnologías que están apareciendo para el desarrollo web.

1.2 Objetivos

El objetivo principal perseguido en esta iniciativa es el de proporcionar a un usuario la posibilidad de establecer una conferencia con cualquier persona de la forma más autónoma posible, es decir, proporcionar una herramienta intuitiva y accesible a todo tipo de públicos y plataformas.

Esta aplicación se va a fundamentar en la utilización únicamente de HTML5 para la representación de todo el contenido multimedia, sin utilizar flash ni ningún otro software.

La consecución de este objetivo, implica un estudio del estado de las aplicaciones web en la actualidad, así como de las herramientas y frameworks utilizados para el desarrollo de los mismos.

Debido a la velocidad con la que van apareciendo nuevas tecnologías en el ámbito de la web, será necesario estudiar periódicamente nuevas alternativas que puedan aumentar el rendimiento de la aplicación, o en algunos casos, simplifiquen el código o la estructura de la misma, lo que suele significar un decremento de los errores que se puedan cometer durante la etapa de desarrollo.

Otro de los objetivos alcanzados con este proyecto es mi desarrollo a nivel profesional. La investigación y desarrollo de la aplicación me han permitido una adquisición de nuevos conocimientos y procedimientos para utilizar durante mi trayectoria profesional.

He intentado utilizar herramientas que se están utilizando en la actualidad, frente a usar los conocimientos adquiridos durante la etapa universitaria, que actualmente pueden estar en desuso. Esto ha dado un valor extra al proyecto y lo ha convertido en un enriquecedor proceso de investigación.

1.3 Fases del desarrollo

Antes de comenzar a esbozar la aplicación fue necesario un tiempo de estudio del estado de las tecnologías utilizadas en el desarrollo web, para comprender cuál es la actual tendencia de desarrollo, así como para examinar lo que está por venir y así establecer unas bases de cara a una futura adaptación.

Una vez realizado este estudio de mercado, es necesario definir los recursos necesarios para desarrollar esta aplicación. Por un lado, se evaluará el hardware mínimo necesario completar el desarrollo. Por otro lado, a partir de los datos obtenidos en la anterior fase de análisis, se seleccionarán las tecnologías que más se adecuen a nuestra aplicación, y en consecuencia, el software necesario para implementar la aplicación. Para reducir costes se utilizará software libre en la medida de lo posible.

Posteriormente a la definición y obtención de recursos, es necesario establecer unas bases para el funcionamiento de la aplicación, así como las funcionalidades que se van a implementar.

A continuación, se empezarán a definir tareas en base a los requisitos y funcionalidades establecidos en la anterior fase.

Inicialmente, las tareas se enfocarán al diseño de la arquitectura de la aplicación, hasta crear un esqueleto robusto. Posteriormente las tareas irán enfocadas a que la aplicación cumpla con los requisitos funcionales.

Según se vayan definiendo tareas, se irán desarrollando y probando de manera dinámica, para que, en caso de no poder llevar a cabo una determinada tarea, definir nuevas tareas de manera rápida y así evitar bloqueos.

Por último, una vez terminado el desarrollo, se procederá a recopilar toda la documentación pertinente para la realización de la memoria del trabajo de fin de grado, para poder comprender el funcionamiento de la aplicación, así como para reflejar todo el trabajo realizado.

1.4 Medios empleados

En este apartado se expone el material utilizado para el desarrollo de la aplicación web.

1.4.1 Hardware

Para poder llevar a cabo este desarrollo es necesario disponer de dos equipos:

El primer equipo será un ordenador personal. En él se programará toda la aplicación y funcionará como servidor para hacer pruebas en local en las primeras fases del desarrollo. Además este equipo posee webcam y micrófono que serán necesarios para probar el correcto funcionamiento de nuestro trabajo.

Va a ser necesario un segundo equipo para alojar la aplicación una vez esté desarrollada. Este segundo equipo va a alojar la base de datos de la aplicación y además se va a usar como servidor para alojar la aplicación. En este caso se utilizará un ordenador del departamento de Telemática.

1.4.2 Software

A continuación se enumera el software utilizado durante el desarrollo del trabajo de fin de grado.

Sublime Text 2: Editor de texto plano capaz de interpretar distintos lenguajes de programación. Con este editor se desarrollará toda la aplicación, tanto la parte cliente como el servidor y la base de datos.

Putty: Cliente SSH necesario para poder acceder al equipo del departamento de Telemática.

WinSCP: Cliente SFTP necesario para poder exportar todos los archivos del ordenador personal al servidor.

Google Chrome/Mozilla Firefox: Navegadores utilizados para depurar las pruebas y hacer pruebas de compatibilidad entre distintas plataformas.

OpenSSL: Utilizado para emitir los certificados de seguridad necesarios para dotar a nuestra aplicación de seguridad. Estos certificados nos permiten aplicar seguridad HTTPS a la aplicación.

Git: Software de control de versiones utilizado para almacenar las distintas versiones del código de la aplicación.

Dropbox: Servicio de almacenamiento de ficheros en la nube utilizado para tener acceso al código de la aplicación desde distintos equipos.

MongoDB: Con este software se creará la base de datos de la aplicación.

Microsoft Word: Utilizado para la redacción de la presente memoria.

1.5 Estructura de la memoria

En esta sección se describe la relación de contenidos tratados en cada uno de los capítulos que componen este documento, con el objetivo de dar al lector una visión general del mismo.

- **Capítulo 1: Introducción y objetivos**

En este capítulo se recoge una visión general del proyecto así como las motivaciones que han llevado a la realización de este trabajo de fin de grado. También se describen las distintas fases por las que ha transcurrido el desarrollo de la aplicación y el material utilizado en las mismas.

- **Capítulo 2: Estado de la cuestión**

En esta sección se recoge un estudio del estado actual de la tecnología de desarrollo web, así como un análisis de la evolución de la web desde los inicios hasta nuestros días.

- **Capítulo 3: Tecnologías utilizadas**

Aquí se detalla la relación de tecnologías utilizadas para desarrollar la aplicación de videoconferencia, así como la justificación de su uso frente a otras tecnologías similares.

- **Capítulo 4: Análisis, diseño e implementación**

Este capítulo cuenta el desarrollo del trabajo de fin de grado, desde la idea, hasta la implementación de la aplicación de videoconferencia.

- **Capítulo 5: Gestión del proyecto**

En este capítulo se recoge la información referida a la planificación del trabajo de fin de grado, así como el presupuesto de la realización del proyecto.

- **Capítulo 6: Conclusiones y trabajos futuros**

El sexto capítulo expondrá las conclusiones alcanzadas tras la realización del trabajo de fin de grado, así como un análisis de los resultados obtenidos y una serie de posibles desarrollos futuros para la aplicación.

- **Glosario:**

Recoge la definición de los términos técnicos utilizados en la redacción de este documento, para que el lector pueda acudir a esta siempre que no entienda el significado de algún término.

- **Bibliografía:**

Listado de referencias consultadas durante la realización de este trabajo ordenadas en orden de aparición en este documento.

Capítulo 2: Estado de la cuestión

En esta sección se va a intentar plasmar toda la información recopilada para poder empezar el desarrollo de nuestra aplicación de videoconferencia. Se estudiará la evolución de la web, su estado actual y las tendencias de desarrollo para poder comprender hacía dónde se encamina la web en el futuro próximo.

2.1 Evolución de la tecnología

Desde que un científico de CERN llamado Tim Berners-Lee (1) inventara la World Wide Web (un sistema de gestión información a partir de hipertextos) en 1989, ésta ha sufrido un gran desarrollo, pasando por distintas fases. A continuación haremos un repaso de los distintos acontecimientos que han marcado la evolución de la web hasta nuestros días.

Inicios de la World Wide Web

En 1990, Berners-Lee había desarrollado todas las herramientas necesarias para trabajar la Web: el protocolo de transferencia de hipertexto (HTTP), el Lenguaje de Marcado de Hipertexto (HTML), el primer navegador Web (llamado WorldWidWeb), el primer servidor de aplicaciones HTTP (CERN httpd), el primer servidor web (http://info.cern.ch/) y las primeras páginas web que describían el proyecto mismo.

Entre 1992 y 1995 comenzaron a aparecer los primero sitios web, principalmente desarrollados por científicos de universidades o laboratorios de física.

A partir de 1996 la mayoría de las compañías importantes comenzaron a tener presencia en la Web, lo que provocó lo que se conoce como el boom de las punto-com.

Primeras páginas web (Web 1.0)

En un principio, las páginas web sólo ofrecían contenidos estáticos, que no daban la posibilidad al usuario de interactuar. Esta etapa se conoce como la web 1.0. En ella los contenidos de las páginas estaban escritos en lenguaje HTML. Básicamente se componía de páginas de sólo lectura, gestionadas por un webmaster que los actualizaba. Cualquier persona podía leer los mensajes, pero sólo los desarrolladores que conocían el lenguaje HTML podían crear y editar páginas web.

Algunas de estas páginas permitían algo de interacción con por parte del usuario a través de formularios sencillos.

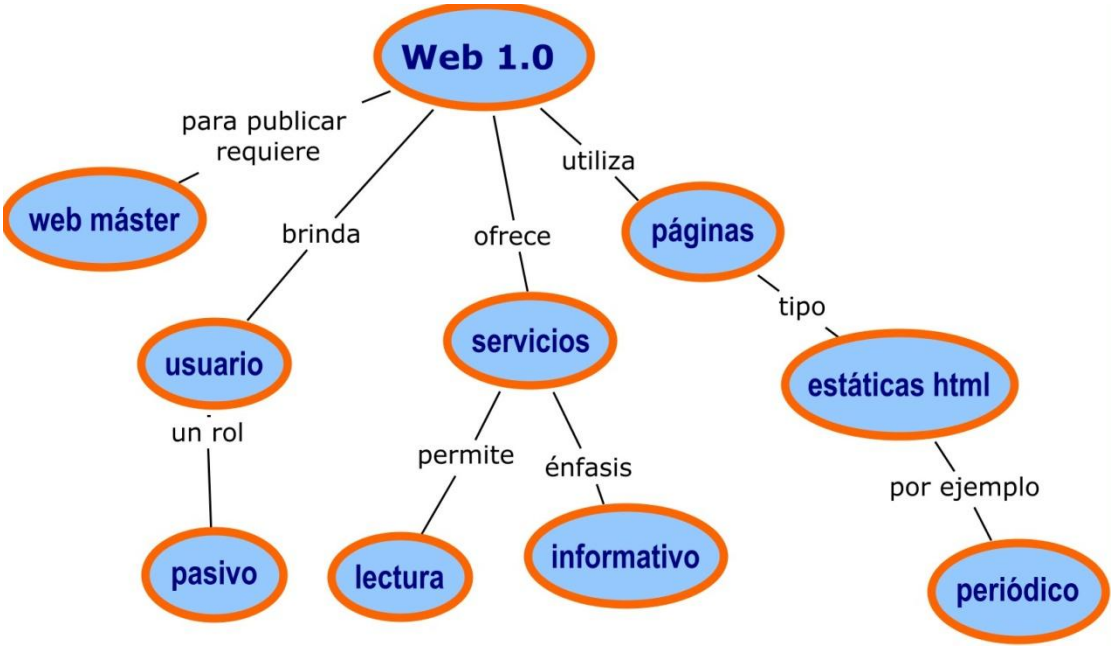

Ilustración 1: Esquema de la Web 1.0

Siguiente fase: Web 2.0

A partir de 2002 surgieron nuevas ideas para compartir información a través de internet como los blogs o las RSS.

Con la aparición del lenguaje Javascript, llegaron las primeras Web 2.0. Este tipo de sitios web se caracterizan por permitir la interacción del usuario con los contenidos ofrecidos. Además la introducción de Flash dotó a la web de contenidos dinámicos mucho más atractivos que los contenidos que se servían en la Web 1.0. Esta fase comprende desde los gestores de correo electrónico (Hotmail), las actuales redes sociales (Facebook y Twitter), los blogs o las wikis que fomentan la colaboración y el intercambio ágil de información entre los usuarios.

Una de las principales características que define a las páginas web de esta etapa es la de la contribución. Frente a la idea de que las web de la anterior etapa fueran gestionadas por una sola persona, en la Web 2.0, aparecen páginas en las que los propios usuarios pueden gestionar los contenidos. Un ejemplo de estos sitios web es YouTube, una página que aloja los vídeos que publican los usuarios. Otro gran ejemplo serían las redes sociales, en las que cada usuario puede agregar el contenido que quiera y compartirlo con otros usuarios.

La inclusión de Javascript en las páginas web hizo posible manejar el comportamiento de las mismas por medio de eventos asociados a elementos del DOM. Además hizo posible establecer comunicación con entre cliente y servidor por

medio de peticiones AJAX, con lo cual se hizo más dinámica la presentación de contenidos.

Ilustración 2: Características de la Web 2.0

La web en la actualidad: Web 3.0

Con la llegada de HTML5 se inició el camino hacia la Web 3.0, una web enfocada a una mayor accesibilidad del usuario. Aparece la web semántica, en la que cada etiqueta describe la información que contiene, de manera que todo el contenido queda mejor indexado de cara a realizar búsquedas utilizando motores de búsqueda.

Esta etapa se caracteriza por la aparición de las aplicaciones web, o lo que es lo mismo, la aparición de herramientas online que permiten realizar las mismas operaciones que antes sólo se podían hacer por medio de programas, que requerían de sistemas operativos complejos para su funcionamiento.

Otra de las características a destacar es la inclusión de contenidos multimedia dentro de HTML5, cosa que hasta antes de aparecer esta tecnología, sólo se podía hacer por medio de Flash. Esto es un hecho verdaderamente importante por dos motivos: el primero es evidente, ya no es necesario aprender dos tecnologías (HTML y Flash); el segundo motivo tiene que ver con que Flash no es libre y necesita de un software de pago para poder desarrollar. (2)

Ilustración 3: Características de la web 3.0

2.2 Tendencias en desarrollo Web

A continuación se detallan algunas de las tendencias utilizadas en el desarrollo de aplicaciones web.

SPI: Single Page Interface

La mayoría de las aplicaciones web que se desarrollan en la actualidad utilizan este paradigma. Éste consiste en crear sitios web con una sola página y con distintos controladores que son los responsables de mostrar los distintos contenidos, a diferencia de los sitios web con varias páginas que se creaban anteriormente (3).

Para hacer esto posible es necesario definir tres elementos que conforman el patrón de arquitectura modelo-vista-controlador:

- **Modelo:** Conjunto de datos que conforman cada una de las secciones de la aplicación. Estos datos son proporcionados por el servidor de la aplicación en

formato XML o JSON.

- **Vista:** Conjunto de plantillas que conforman la interfaz de la aplicación. Son las responsables de la forma en que se representarán los datos del modelo.

- **Controlador:** Es el responsable de responder a los eventos que desencadena el usuario con su navegación para realizar las tareas necesarias y actualizar los datos presentados al usuario. También será el responsable de la comunicación con el servidor.

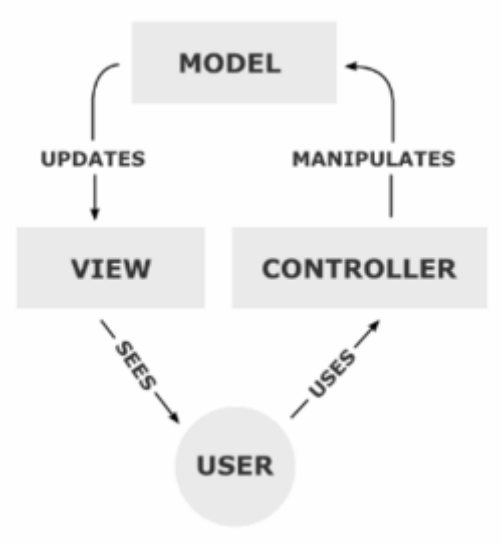

Ilustración 4: Arquitectura Modelo-Vista-Controlador

En este tipo de arquitectura, cuando el usuario interactúa con la aplicación, se desencadenan una serie de eventos que interpreta el controlador para actualizar los datos del modelo que van a ser necesarios para responder a las necesidades del usuario. La actualización de los datos del modelo provoca un cambio en los datos mostrados en la vista.

Por ejemplo, imaginemos que un usuario quiere darse de alta en una red social implementada con esta tecnología. Cuando el usuario rellene el formulario con sus datos y pulse el botón para registrarse, el controlador de la aplicación llevará a cabo una serie de tareas como pueden ser: la validación de los datos del usuario, el registro de los datos del usuario en la base de datos de la aplicación, etc. Una vez realizadas estas tareas, añadirá al modelo los datos necesarios para responder al usuario y pintará por pantalla la vista correspondiente incluyendo los datos de la respuesta: fallos en la validación de datos, mensajes de enhorabuena, etc.

Javascript como motor principal de la web

Con todo lo comentado hasta ahora podemos deducir algunas cosas sobre la Web. Sabemos que las páginas/aplicaciones web están escritas en lenguaje HTML y que utilizan Javascript para interactuar con el usuario, pero, ¿qué hay detrás de las páginas web? ¿Quién es el responsable de servir toda la información que contiene una web?

Pues bien, ya se ha mencionado anteriormente la existencia de un **servidor**, y éste es el responsable de servir los contenidos para cada vista de nuestra aplicación. Hasta hace relativamente poco, la comunicación con el servidor se realizaba a partir de aplicaciones escritas en otros lenguajes de programación como php, ASP, etc. En caso de aplicaciones de gran complejidad el servidor suele estar implementado sobre la tecnología Java.

En 2009 aparece en escena Node.js (4), un entorno de programación en la capa del servidor basado en el lenguaje Javascript. Este hecho posiblemente marque un antes y un después en el desarrollo de aplicaciones web, pues ya no es necesario que el desarrollador de aplicaciones tenga que tener conocimientos de distintas tecnologías, sino que todo se reduce a que el desarrollador conozca sólo el lenguaje Javascript.

Node.js además incorpora varios módulos básicos que proporcionan un nivel de abstracción para ejecutar funciones de red, y además permite usar módulos desarrollados por terceros, cosa que simplifica la tarea de desarrollo bastante.

El uso de Node.js está siendo cada vez más extendido en el mundo del desarrollo web. De hecho hay algunas empresas importantes que se están migrando de Node.js como es el caso de PayPal (5).

Aplicaciones multiplataforma

Anteriormente a la aparición de los smartphones, la mayor parte de los accesos a la web se hacía a través de ordenadores personales, por lo que cuando se diseñaba un sitio web no se tenía en cuenta la relación de aspecto, puesto que la resolución de los monitores no variaba en demasía.

Según iban aumentado los accesos a la web a través de dispositivos móviles, algunas empresas empezaron a sacar versiones de su aplicación para móvil.

A raíz de la proliferación del uso de smartphones, han cambiado los hábitos de los clientes de la web y su forma de navegar. Esto supone un reto para los desarrolladores de aplicaciones, pues han de adaptar sus contenidos para que puedan

verse correctamente desde distintos dispositivos con distinta resolución.

Es por esto, que la tendencia es crear webs siguiendo la filosofía del *responsive design* (6), o lo que es lo mismo, páginas que adapten su contenido a la resolución de pantalla del dispositivo desde el que se accede.

En los últimos años han aparecido algunos frameworks de CSS que se encargan de la labor de adaptar los contenidos a las distintas resoluciones de pantalla, algunos ejemplos de estos frameworks son Bootstrap o Foundation.

También es usual encontrarnos con aplicaciones para sistemas operativos móviles como Android o iOS implementadas sobre HTML5. Esto es posible gracias a un tipo de frameworks capaces de encapsular aplicaciones web en aplicaciones nativas de estos sistemas operativos. Un ejemplo bastante representativo de este tipo de frameworks es PhoneGap.

Accesibilidad

Por último, me gustaría destacar también la consciencia que se ha adquirido en los últimos años para desarrollar aplicaciones accesibles. Este tipo de aplicaciones se centran en hacer llegar sus contenidos a personas con discapacidades.

De hecho recientemente se han ido estableciendo una normativa para la accesibilidad a los contenidos en la web (7). En España se aprobó en julio de 2012 la Norma UNE 139803:2012. Requisitos de Accesibilidad para contenidos en la web.

Desde la W3C se propone la iniciativa WAI-ARIA (8) en la que se aplican algunos atributos a las etiquetas HTML para definir los roles de las mismas, añadir información para el lector de pantalla (especialmente centrado en las personas con discapacidades visuales), permitir la navegación por los contenidos a partir del teclado, etc.

Utilización de frameworks de desarrollo

Anteriormente se han mencionado algunos frameworks de desarrollo. Estos son herramientas desarrolladas por terceros que aportan un nivel de abstracción para realizar alguna tarea. Son como una especie de librerías.

Algunos ejemplos de estas herramientas son:

Jade: Este framework permite escribir páginas HTML utilizando únicamente selectores CSS3 y tabulaciones.

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
         bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
         bar(1 + 5)
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
      </p>
    </div>
  </body>
</html>
```

Ilustración 5: Comparativa de una página escrita con Jade y con HTML

Sass: Es una herramienta para la definición de estilos. Permite la agrupación de estilos y que unos componentes hereden los estilos de otros. Además permite crear variables y funciones (mixins) para utilizar en distintos archivos.

Express.js: Este es un módulo de Node.js que permite la creación del servidor de una aplicación web de una forma bastante sencilla.

2.3 Estudio del mercado de las videoconferencias

Aplicaciones de escritorio

Antes de que se produjera el boom del desarrollo de aplicaciones web, existían algunos programas o aplicaciones de escritorio que permitían hacer videoconferencias a través de internet. Destaca entre todos éstos programas Skype, la aplicación de videoconferencia por excelencia.

Estas aplicaciones necesitaban de un software que el usuario tenía que instalar en su ordenador para obtener toda la información de los dispositivos multimedia, procesarla y enviarla por la red hasta el usuario destino de la llamada.

Las videollamadas utilizaban el protocolo VoIP y fundamentalmente SIP como protocolo de señalización, y RTC y RTCP como protocolos de comunicación.

Primeras aplicaciones web: WebEx

No tardaron en aparecer las primeras aplicaciones online para realizar videoconferencias. Un ejemplo de estas aplicaciones es la plataforma WebEx, una plataforma desarrollada por Cisco Systems enfocada fundamentalmente a la realización de videoconferencias online para empresas.

WebEx es una plataforma de pago que permite crear conferencias por Internet en las que los usuarios, además de verse y hablar los unos a los otros, pueden compartir datos y archivos de manera segura.

El problema de este tipo de aplicaciones es, que aunque no requieren de un software propio para funcionar, si que necesitan algunas extensiones o plugins instalados en el ordenador de los usuarios. Por lo que antes de poder realizar la videoconferencia online, vamos a tener que instalar estos complementos en nuestro ordenador.

WebRTC

En mayo de 2011, Google lanzó WebRTC (9) (Web Real-Time Communication), una API para la comunicación a tiempo real basada en el navegador. Esta API es de código abierto con licencia BSD y está estandarizada por la IETF y el W3C.

De momento WebRTC se encuentra en un estado avanzado de desarrollo y estandarización y es soportada por los navegadores Chrome, Firefox y Opera, aunque la mayoría de los fabricantes de infraestructura de red han anunciado el soporte de WebRTC para sus productos.

La mayor ventaja que propone WebRTC es que forma parte de HTML5 y no necesita de la instalación de ningún plugin propietario adicional.

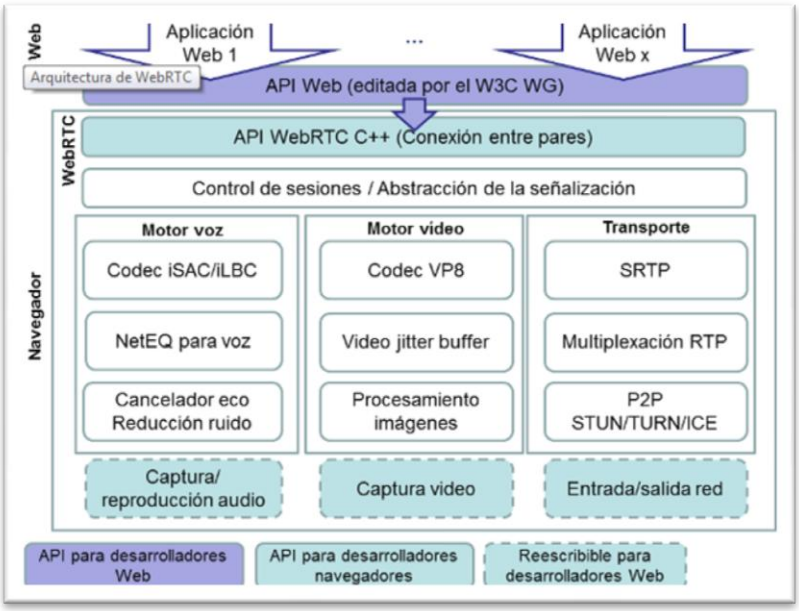

Ilustración 6: Arquitectura de WebRTC

Una de las características de WebRTC en comparación con los tradicionales sistemas de sistemas de comunicación IP, es que ignora el protocolo de señalización, para permitir a los fabricantes de sistemas VoIP emplear esta API independientemente del protocolo de señalización utilizado (10).

WebRTC integra los codecs de audio G.711 y Opus. Este último, a demás de ser gratuito, es de código abierto, y además, es uno de los mejores codecs de audio que existen.

Para el vídeo, WebRTC utiliza VP8 en lugar de usar H.264, que es el más popular, debido a los costes de licencia de MPEG. El único problema de VP8 es que es nuevo, por lo que no tiene soporte nativo en la mayoría de los chipsets actuales.

Característica	VoIP	WebRTC
Señalización	SIP (principalmente) y H.323, en la mayoría de los casos	Sin definir
Medios	RTC/RTCP	
Codecs de voz	Serie G.7xxx (principalmente)	G.711 y Opus
Codecs de vídeo	H.263, H.264	VP8
Seguridad de los medios	SRTP en SIP, H.235 en H.323	SRTP
NAT Transversal	STUN/TURN/ICE en SIP, H.450.x en H.323	STUN/TURN/ICE

Tabla 2: Características de WebRTC frente a VoIP

Algunas de las compañías que ofrecen sistemas de videoconferencia ya han anunciado que van a comenzar a utilizar WebRTC, y es cuestión de tiempo que todas las videoconferencias online se realicen utilizando esta API. (11)

Capítulo 3: Tecnologías utilizadas

En el presente apartado se va a tratar la relación de todas las tecnologías con las que se ha trabajado para conseguir desarrollar la aplicación de videoconferencia, así como la justificación de su uso en lugar de otras tecnologías similares.

En nuestro caso, no existe un marco que regule el desarrollo de aplicaciones web, y además toda la tecnología que se utilizará para el desarrollo de este trabajo de fin de grado es libre, por lo que no habrá ningún problema de derechos.

Se van a definir dos grupos de tecnologías en función de si son utilizadas en la parte cliente de la aplicación o en la parte servidor. La toma de decisiones a la hora de escoger las tecnologías siempre se ha basado en los resultados obtenidos en el estudio de las tecnologías visto en el anterior capítulo de esta memoria.

3.1 Cliente de Videoconferencia

Después del análisis de la tecnología llevado a cabo en el anterior capítulo, somos capaces de empezar a definir la arquitectura de nuestra aplicación de videoconferencia, y las herramientas necesarias para darle forma.

Dado que se trata de una aplicación de videoconferencia, se decide implementarla usando la API de WebRTC.

Además, se tomó la decisión de implementar una aplicación web que siga la especificación de las SPI, por lo que se necesitará usar un tipo de tecnología que soporte este modelo de aplicaciones web. En nuestro caso, se eligió el framework de Javascript Backbone.js (12), que nos permite separar los componentes modelo-vista-controlador.

Existen otros frameworks para desarrollar aplicaciones SPI como Angular.js (13) y Polymer.js (14), que aunque están en desarrollo, se están comenzando a usar cada vez más. Pero se tomó la decisión de utilizar Backbone por el conocimiento previo de esta tecnología.

Por último, se pensó en hacer una aplicación con diseño responsive, para lo que fue necesario el uso de Bootstrap.

3.1.1 HTML5

La elección de esta tecnología es bastante obvia dado que para poder usar el API de WebRTC es necesario que el lenguaje de nuestra página sea HTML5.

Además, la versión 5 de este lenguaje, nos proporciona una serie de etiquetas para manejar los contenidos multimedia (en nuestro caso vamos a necesitar mostrar audio y vídeo).

Con HTML5 se van a generar las plantillas que conformarán las vistas de nuestra aplicación SPI.

3.1.2 Javascript

El uso de esta tecnología es indispensable a la hora de desarrollar aplicaciones web, pues es la responsable de manejar la interacción con el usuario. En nuestra aplicación, nos proporcionará las capacidades para crear los controladores.

Utilizaremos los siguientes frameworks de Javascript en nuestra aplicación:

jQuery (15)

Este framework nos proporciona la capacidad de interactuar con el DOM de forma sencilla. En nuestra aplicación, se ha utilizado para acceder a los contenidos de las etiquetas HTML.

Además, nos proporciona la capacidad de comunicar nuestro cliente con el servidor por medio de AJAX, por lo que para establecer la comunicación con el servidor de la aplicación también utilizaremos jQuery.

Underscore.js (16)

Esta librería nos aporta gran cantidad de funciones para tratar conjuntos de datos como objetos JSON y listas, por lo que en nuestra aplicación nos permitirá manejar los datos del modelo que serán representados en la vista.

Además, nos proporciona funciones para insertar los datos del modelo dentro del código HTML de las plantillas, e incluso, nos da la opción de añadir nuestras propias funciones.

Utilizaremos la función *_.template* para introducir los datos del modelo en nuestras plantillas, y la función *_.mixin* para añadir nuestras propias funciones.

Backbone.js

Esta es la librería que nos permite implementar el paradigma Modelo Vista Controlador en nuestra aplicación. Es la responsable de crear los controladores y añadir el modelo a cada vista.

Backbone nos proporciona entre otras cosas las vistas (Views) y los modelos (Models). Para Backbone, una vista implementa al controlador. En la View de Backbone se definen los datos referentes a la vista, tanto el template que representará los datos, como el contenedor dónde mostrar la vista. Además cada vista de Backbone llevará asociado un modelo.

El modelo de Backbone es un objeto de tipo JSON en el que se incluyen todos los datos (incluso funciones) asociados a una vista.

En la vista de Backbone se definen también todos los eventos asociados a elementos del DOM y las funciones que se ejecutarán en respuesta a estos eventos.

Por último, Backbone nos proporciona también un router, es decir, una pieza cuyo objetivo es establecer una correspondencia entre las vistas y las rutas de las mismas.

WebRTC

Esta API es la que se va a ocupar de capturar tanto el audio como el vídeo de cada usuario, procesarlo y enviarlo hasta el otro lado.

WebRTC nos proporciona las funciones *getUserMedia*, para obtener y procesar el flujo de datos de los dispositivos multimedia del usuario, y *RTCPeerConection*, que será la encargada de establecer el envío de estos datos multimedia entre usuarios.

Require.js (17)

Con Require se maneja toda la gestión de dependencias de la aplicación. Nos sirve para establecer una jerarquía entre los distintos scripts.

Con esta librería vamos a definir las dependencias de cada uno de nuestros ficheros Javascript, y además, exportar nuestras dependencias como variables. Para comprender cómo funciona nos podemos apoyar en la siguiente imagen:

```
require.config({
  shim: {
    'socketio': {
      deps: [],
      exports: 'io'
    },
    'underscore': {
      deps: [],
      exports: '_'
    },
    'backbone': {
      deps: ['jquery', 'underscore'],
      exports: 'Backbone'
    },
```

Ilustración 7: Ejemplo de uso de Require.js

En este caso, vemos que para cada librería se definen en *deps*, el listado de dependencias, y en *exports*, la variable que va a representar a cada librería. Vemos que en el caso de underscore se exporta como "_", esto significa que cada vez que se utilice este caracter, la aplicación lo va a tratar como una referencia a la librería de underscore. Además, vemos que Backbone requiere de underscore y jQuery para poder funcionar.

Socket.io (18)

Esta librería permite establecer comunicaciones a tiempo real entre usuarios por medio de sockets.

Es la pieza que va a establecer la comunicación entre los usuarios de la aplicación para que puedan realizar la videoconferencia.

3.1.3 CSS3

De la mano de HTML5, apareció CSS3, que permite agregar hojas de estilo para maquetar el contenido HTML. Aunque nuestra aplicación web hace uso de Bootstrap, también se utiliza esta tecnología para modificar los estilos de los componentes predefinidos por Bootstrap.

Bootstrap (19)

Este framework define una serie de estilos por medio del uso de clases en los elementos HTML, por lo que basta con añadir estas clases a los elementos del nuestra aplicación para dotarlos de estilo.

También define una serie de componentes como tablas, barras de navegación, desplegables, etc. Para que estos componentes funcionen, es necesario añadir a

nuestro proyecto, además del css de Bootstrap, el fichero Javascript responsable de su funcionamiento.

Se optó por el uso de este framework, además de para que nuestra aplicación tenga diseño *responsive*, para no perder demasiado tiempo en tareas de maquetación.

3.2 Servidor de la aplicación

Para el servidor, se tomó la decisión de implementarlo con Node.js frente a la utilización de un Apache desarrollado en Java, por los siguientes motivos:

- Utilización de Javascript como único lenguaje de programación tanto en el Front-end, como en el Back-end.
- Es una herramienta que está tomando mucha importancia en desarrollo Back-end.
- Node.js ofrece la posibilidad de utilizar distintos módulos que implementan tareas determinadas, tanto módulos básicos que incluye este framework, como módulos desarrollados por terceros.

Por otro lado, se eligió utilizar una base de datos MongoDB (20) para almacenar los datos de los usuarios, frente a usar una base de datos relacional SQL.

Este tipo de base de datos, permite almacenar directamente objetos, no es necesario adaptar los datos para seguir una estructura de tablas. Dado que nuestra aplicación maneja los datos con objetos JSON, la elección de esta tecnología tiene bastante sentido.

3.2.1 Node.js

Como ya se ha comentado, nuestra aplicación de videoconferencia va a utilizar un servidor desarrollado con la tecnología Node.js.

Para la gestión de los distintos módulos de Node.js, se utiliza la herramienta NPM, que permite la descarga y utilización de los mismos. En concreto, se van a utilizar los siguientes módulos en nuestra aplicación:

Express.io (21)

Este módulo es el resultado de la combinación de dos módulos de Node.js:

- **Express.js:** Módulo capaz de crear un servidor de manera sencilla.

- **Socket.io:** Este módulo permite que la aplicación pueda establecer comunicaciones en tiempo real.

Https

Este módulo es uno de los que proporciona Node.js, y permite la creación de un servidor HTTPS a partir de los certificados de seguridad necesarios.

Fs

Este módulo básico permite el acceso del servidor al sistema de ficheros. Con él, podremos acceder a la información de algunos ficheros, por ejemplo, para obtener la información de los certificados de nuestro servidor HTTPS.

Path

Otro módulo de los que incluye Node.js. Éste, permite hacer modificaciones en las rutas de los ficheros, y se utiliza para definir la ruta de los ficheros estáticos servidos por nuestra aplicación.

bcrypt-nodejs (22)

Se utilizó este módulo para encriptar los datos sensibles del usuario a la hora de almacenarlos en la base de datos de la aplicación.

3.2.2 MongoDB

En nuestra aplicación necesitamos almacenar los datos de los usuarios, ya que será requisito, que los usuarios estén dados de alta para poder realizar videoconferencias.

Se eligió esta tecnología porque nos ofrece la posibilidad de almacenar los datos de usuario de forma directa, sin necesidad de crear distintas tablas para almacenar todos los datos del usuario.

MongoDB, permite almacenar los datos en formato clave-valor, esto significa que podremos almacenar los datos del modelo de la aplicación directamente, sin aplicarles ningún formato.

Mongoose.js (23)

Mongoose es un framework que nos ayuda a manejar las bases de datos

Mongo. Mongoose ofrece un API bastante extenso que nos permite comunicarnos con nuestra base de datos.

En nuestro desarrollo de la aplicación, nos hemos encontrado con dos formas en las que Mongoose obtiene los datos de la base de datos. Si no se le aplica ningún tipo de lógica, obtenemos todos los datos del usuario en forma de modelo. Pero también, podemos obtener los datos en formato JSON, y así poder manejarlos y utilizar sólo los datos que necesitemos en cada caso. En el caso de esta aplicación, va a haber datos sensibles del usuario, como su contraseña, que no vamos a devolver en el modelo a la vista.

Capítulo 4: Análisis, diseño e implementación

En este capítulo se hablará de todo el proceso de desarrollo de este trabajo de fin de grado, desde el análisis inicial, hasta la implementación.

Para facilitar la lectura, se ha dividido este capítulo en tres partes diferenciadas. En primer lugar se abordará el análisis del proyecto, es decir, los requisitos que debe cumplir nuestra aplicación. A continuación se detalla la fase de diseño de la aplicación, dónde se verá la arquitectura de la aplicación. Por último, veremos la fase de implementación, en la que se detallarán los aspectos más importantes del desarrollo de la aplicación de videoconferencia.

4.1 Análisis

En este punto, veremos cuál es el funcionamiento básico de la aplicación, y quedará definido su comportamiento a través de unos requisitos y casos de uso.

Para evitar que la lectura de este apartado sea pesada, se van a definir tanto los requisitos como los casos de uso por medio de tablas.

4.1.1 Requisitos

En esta sección definiremos los requisitos que ha de cumplir nuestra aplicación de videoconferencia. Para ello utilizaremos el siguiente estilo de tabla:

Identificador:	**Nombre:**
Descripción:	
Importancia:	

Tabla 3: Formato de tabla de requisitos

Para cada uno de los requisitos se adjuntará la siguiente información:

- **Identificador:** Código que identifica al requisito en cuestión. Este código tendrá siguiente formato: “REQ-XX”, donde XX será un número entero.
- **Nombre:** Nombre del requisito, que debe ser auto explicativo, es decir, sólo con leer el nombre deberíamos ser capaces de entender el requisito.
- **Descripción:** Breve descripción que acompaña al nombre del requisito. Sirve para aportar más información sobre el requisito.
- **Importancia:** Grado de importancia del requisito, que servirá para determinar una prioridad a la hora de implementar las tareas asociadas.

A continuación se muestra la lista con los requisitos que ha de cumplir la aplicación de videoconferencia:

Identificador:	**REQ-01**	**Nombre:**	**Aplicación HTML5**
Descripción:	La aplicación debe estar escrita en HTML5 y no debe hacer uso de ningún plugin adicional.		
Importancia:	Alta		

Tabla 4: REQ-01. Aplicación HTML5

Identificador:	**REQ-02**	**Nombre:**	**Servidor en Node.js**
Descripción:	El servidor de la aplicación se implementará en lenguaje Javascript usando Node.js como motor.		
Importancia:	Alta		

Tabla 5: REQ-02. Servidor en Node.js

Identificador:	**REQ-03**	**Nombre:**	**Aplicación SPI**
Descripción:	La aplicación seguirá el modelo SPI, para ello se hará uso de Backbone.js.		
Importancia:	Alta		

Tabla 6: REQ-03. Aplicación SPI

Identificador:	**REQ-04**	**Nombre:**	**Registro de usuarios**
Descripción:	Para poder establecer sesiones de videoconferencia los usuarios han de registrarse en la plataforma.		
Importancia:	Alta		

Tabla 7: REQ-04. Registro de usuarios

Identificador:	**REQ-05**	**Nombre:**	**Almacenamiento de datos de usuario**
Descripción:	Se almacenarán datos del usuario con el objetivo de establecer sesiones. Para ello se empleará una base de datos MongoDB.		
Importancia:	Alta		

Tabla 8: REQ-05. Almacenamiento de datos de usuario

Identificador:	**REQ-06**	**Nombre:**	**Cifrado de datos sensibles**
Descripción:	Los datos sensibles del usuario que se guarden en la base de datos deberán estar cifrados.		
Importancia:	Alta		

Tabla 9: REQ-06. Cifrado de datos sensibles

Identificador:	**REQ-07**	**Nombre:**	**Seguridad en las sesiones**
Descripción:	La aplicación tiene que tener alto nivel de seguridad para que nadie pueda filtrar las conversaciones de los usuarios. Para ello se implementará un servidor HTTPS.		
Importancia:	Alta		

Tabla 10: REQ-07. Seguridad en las sesiones

Identificador:	**REQ-08**	**Nombre:**	**Comunicación segura cliente-servidor**
Descripción:	Todas las peticiones que se hagan al servidor han de incluir un token CSRF (24) en la cabecera. Este token estará incluido en una etiqueta meta de la cabecera HTML de la aplicación.		
Importancia:	Alta		

Tabla 11: REQ-08. Comunicación segura cliente-servidor

Identificador:	**REQ-09**	**Nombre:**	**Duración de las sesiones**
Descripción:	Las sesiones permanecerán abiertas durante un periodo de un día de duración, a no ser que el usuario se desconecte explícitamente.		
Importancia:	Alta		

Tabla 12: REQ-09. Duración de las sesiones

Identificador:	**REQ-10**	**Nombre:**	**Captura de flujos multimedia**
Descripción:	La aplicación ha de ser capaz de capturar y procesar los flujos de los dispositivos multimedia del usuario, así como de reproducirlos de manera local.		
Importancia:	Alta		

Tabla 13: REQ-10. Captura de flujos multimedia

Identificador:	**REQ-11**	**Nombre:**	**Establecimiento de videoconferencia**
Descripción:	La aplicación ha de ser capaz de establecer una sesión de videoconferencia entre dos usuarios.		
Importancia:	Alta		

Tabla 14: REQ-11. Establecimiento de videoconferencia

Identificador:	**REQ-12**	**Nombre:**	**Fin de videoconferencia**
Descripción:	La aplicación ha de ser capaz de finalizar una llamada, así como de parar la captura de flujos multimedia.		
Importancia:	Alta		

Tabla 15: REQ-12. Fin de videoconferencia

Identificador:	REQ-13	Nombre:	Cerrar sesión
Descripción:	Un usuario debe ser capaz de cerrar sesión siempre que lo desee.		
Importancia:	Alta		

Tabla 16: REQ-13. Cerrar sesión

Identificador:	REQ-14	Nombre:	Nombre de usuario único
Descripción:	No puede haber más de un usuario con el mismo nombre de usuario.		
Importancia:	Alta		

Tabla 17: REQ-14. Nombre de usuario único

4.1.2 Casos de uso

En este apartado se definen los casos de uso que tiene que ha de implementar nuestra aplicación de videoconferencia.

Cada uno de ellos vendrá representado en una tabla con la siguiente estructura:

Identificador:	Nombre:
Descripción:	
Precondiciones:	
Resultado esperado:	
Prioridad:	

Tabla 18: Formato de tabla para casos de uso

Como se contempla en la figura anterior, la estructura es similar a la estructura de las tablas de definición de requisitos. En este caso los campos representan:

- **Identificador:** Código que identifica al caso de uso, este código tendrá siguiente formato: “CU-XX”, donde XX será un número entero.
- **Nombre:** Nombre del caso de uso que debe ser auto explicativo.
- **Descripción:** Breve descripción que acompaña al nombre del caso de uso. Sirve para aportar más información sobre la operación.
- **Precondiciones:** Sitúa el contexto del caso de uso, es decir, las condiciones que se han de cumplir para poder evaluar esta operación.
- **Resultado esperado:** Indica la respuesta de la aplicación ante el comportamiento reflejado en la descripción del caso de uso.
- **Prioridad:** Denota la importancia de la tarea a realizar.

Para nuestra aplicación de videoconferencia se definieron los siguientes casos de uso:

Identificador:	**CU-01**	**Nombre:**	**Acceso a la aplicación**
Descripción:	Un usuario accede a la aplicación.		
Precondiciones:	El usuario no está logado.		
Resultado esperado:	Se muestra una pantalla de bienvenida que invita al usuario a logarse o a registrarse.		
Prioridad:	Alta		

Tabla 19: CU-01. Acceso a la aplicación

Identificador:	**CU-02**	**Nombre:**	**Acceso a la aplicación usuario logado**
Descripción:	Un usuario accede a la aplicación.		
Precondiciones:	El usuario está logado.		
Resultado esperado:	Se muestra la home de la aplicación, donde aparece el nombre del usuario y el listado de contactos disponible.		
Prioridad:	Alta		

Tabla 20: CU-02. Acceso a la aplicación usuario logado

Identificador:	**CU-03**	**Nombre:**	**Registro en la aplicación**
Descripción:	El usuario quiere darse de alta.		
Precondiciones:	Desde la pantalla de bienvenida, el usuario selecciona la opción para darse de alta en la plataforma.		
Resultado esperado:	Se redirige al usuario a la pantalla de nuevas altas, en la que se muestra un formulario donde introducir sus datos.		
Prioridad:	Alta		

Tabla 21: CU-03. Registro en la aplicación

Identificador:	**CU-04**	**Nombre:**	**Solicitud de registro incompleta**
Descripción:	El usuario envía el formulario de registro.		
Precondiciones:	Desde la pantalla de registro, el usuario no rellena alguno de los campos.		
Resultado esperado:	Se mostrará un mensaje de error informando de que todos los datos son obligatorios.		
Prioridad:	Media		

Tabla 22: CU-04. Solicitud de registro incompleta

Identificador:	**CU-05**	**Nombre:**	**Solicitud de registro usuario repetido**
Descripción:	El usuario envía el formulario de registro		
Precondiciones:	Desde la pantalla de registro, el usuario elige un nombre de usuario existente.		
Resultado esperado:	Se informará al usuario de que el nombre de usuario ya existe para que elija otro.		
Prioridad:	Alta		

Tabla 23: CU-05. Solicitud de registro usuario repetido

Identificador:	**CU-06**	**Nombre:**	**Solicitud de registro correcta**
Descripción:	El usuario envía el formulario de registro		
Precondiciones:	Desde la pantalla de registro, el usuario envía una solicitud da alta con los datos correctos.		
Resultado esperado:	Se almacenarán los datos del usuario en la base de datos de la aplicación, se iniciará sesión y se mostrará la home de la aplicación.		
Prioridad:	Alta		

Tabla 24: CU-06. Solicitud de registro correcta

Identificador:	**CU-07**	**Nombre:**	**Inicio de sesión incompleto**
Descripción:	El usuario envía el formulario inicio de sesión.		
Precondiciones:	Desde la pantalla de bienvenida, el usuario intenta acceder a la aplicación sin introducir el usuario o la contraseña.		
Resultado esperado:	Se mostrará un mensaje de error indicando que los campos son obligatorios.		
Prioridad:	Alta		

Tabla 25: CU-07. Inicio de sesión incompleto

Identificador:	**CU-08**	**Nombre:**	**Inicio de sesión con usuario inválido**
Descripción:	El usuario envía el formulario inicio de sesión.		
Precondiciones:	Desde la pantalla de bienvenida, el usuario intenta acceder a la aplicación con un usuario no existente en la base de datos.		
Resultado esperado:	Se mostrará un mensaje de error indicando que el usuario no existe.		
Prioridad:	Media		

Tabla 26: CU-08. Inicio de sesión con usuario inválido

Identificador:	**CU-09**	**Nombre:**	**Inicio de sesión con contraseña inválida**
Descripción:	El usuario envía el formulario inicio de sesión.		
Precondiciones:	Desde la pantalla de bienvenida, el usuario intenta acceder a la aplicación con un usuario existente y una contraseña incorrecta.		
Resultado esperado:	Se mostrará un mensaje de error indicando que la contraseña es incorrecta.		
Prioridad:	Media		

Tabla 27: CU-09. Inicio de sesión con contraseña inválida

Identificador:	**CU-10**	**Nombre:**	**Inicio de sesión correcto**
Descripción:	El usuario envía el formulario inicio de sesión.		
Precondiciones:	Desde la pantalla de bienvenida, el usuario intenta acceder a la aplicación con un usuario y contraseña correctos.		
Resultado esperado:	Se establecerá una nueva sesión y se redirigirá a la home.		
Prioridad:	Alta		

Tabla 28: CU-10. Inicio de sesión correcto

Identificador:	**CU-11**	**Nombre:**	**Menú de configuración**
Descripción:	Hacer click en el menú de configuración de la barra de navegación.		
Precondiciones:	El usuario ha de estar logado.		
Resultado esperado:	Se despliega el menú de configuración con distintas opciones de navegación y una opción para cerrar sesión		
Prioridad:	Baja		

Tabla 29: CU-11. Menú de configuración

Identificador:	**CU-12**	**Nombre:**	**Cerrar sesión**
Descripción:	Hacer click en la opción “Cerrar sesión” del menú de configuración.		
Precondiciones:	El usuario ha de estar logado.		
Resultado esperado:	Se ha de cerrar la sesión del usuario y redirigir a la pantalla de bienvenida.		
Prioridad:	Alta		

Tabla 30: CU-12. Cerrar sesión

Identificador:	**CU-13**	**Nombre:**	**Videollamada a usuario offline/ocupado**
Descripción:	Hacer click en un usuario de la lista no disponible.		
Precondiciones:	El usuario ha de estar logado en la home.		
Resultado esperado:	La aplicación no debe hacer nada.		
Prioridad:	Baja		

Tabla 31: CU-13. Videollamada a usuario offline/ocupado

Identificador:	**CU-14**	**Nombre:**	**Videollamada a usuario online**
Descripción:	Hacer click en un usuario de la lista conectado.		
Precondiciones:	El usuario ha de estar logado en la home.		
Resultado esperado:	Se debe iniciar el proceso de videoconferencia mostrando el video local.		
Prioridad:	Alta		

Tabla 32: CU-14. Videollamada a usuario online

Identificador:	**CU-15**	**Nombre:**	**Finalizar videollamada**
Descripción:	Hacer click en el botón “Finalizar llamada”		
Precondiciones:	El usuario ha de estar realizando una videoconferencia.		
Resultado esperado:	Se debe detener el proceso de videoconferencia eliminando tanto el video local, como el remoto.		
Prioridad:	Alta		

Tabla 33: CU-15. Finalizar videollamada

4.2 Diseño

En este apartado se hablará sobre la fase de diseño de la aplicación. Cabe distinguir distintos tipos de diseño. Desde el punto de vista estético, hablaremos del diseño de la interfaz de la aplicación. Por otro lado, tenemos el diseño que nos definen los distintos requisitos y casos de uso, a este diseño lo denominaremos diseño lógico o arquitectura de la aplicación.

4.2.1 Estética de la aplicación

En este apartado se darán unas pautas para el diseño de la aplicación. Se ha optado por hacer una aplicación con una interfaz bastante sencilla e intuitiva para facilitar su correcto uso.

Tal y como se ha comentado en anteriores secciones de este documento, para facilitar la maquetación de la aplicación se utilizará Bootstrap, lo cual, añade el valor añadido de hacer que la aplicación tenga responsive design. Para poder

customizar los elementos que proporciona Bootstrap, se creará un fichero css con el objetivo de no modificar la librería.

A la vista de los requisitos y los casos de uso vistos en el punto anterior, se intuyen dos tipos de pantalla, en función de si el usuario está logado o no. En ambos casos se dividirá la pantalla en tres espacios diferenciados:

Encabezado: En caso de pantallas en las que el usuario no se encuentre logado en la aplicación, se mostrará el título de la sección. Para usuarios logados, el encabezado se sustituirá por una barra de navegación con los accesos a las distintas secciones.

Contenedor principal: En esta división se muestra el contenido principal de la sección a mostrar. Esta sección ocupará tres cuartas partes del ancho de la aplicación.

Contenedor secundario: En esta sección se mostrará contenido menos importante. Este contenedor se sitúa a la derecha del contenedor principal, complementando la información de éste. Ocupará la cuarta parte del ancho de la pantalla.

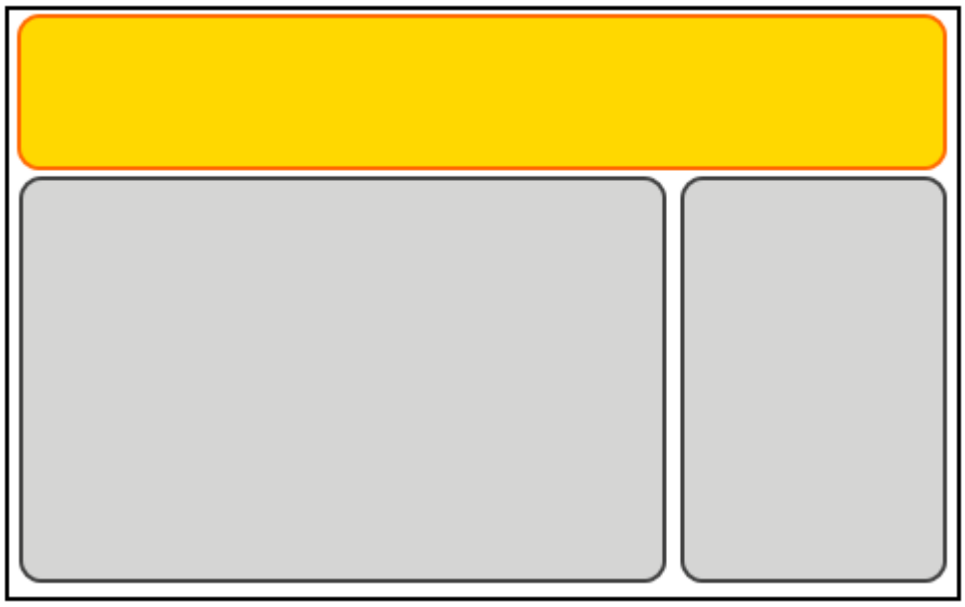

Ilustración 8: Definición de la interfaz de la aplicación

4.2.2 Arquitectura lógica

En esta sección se va a definir la estructura que va a tener la aplicación desde el punto de vista lógico. Esta estructura va a estar determinada por los requisitos de la aplicación, así como por los casos de uso anteriormente definidos.

Uno de los principales requisitos que esta aplicación ha de cumplir, es que debe consistir en una única página (modelo SPI), por lo que vamos a tener que definir una serie de estados o vistas para mostrar las distintas secciones de la aplicación.

En el capítulo 3 de este documento se mencionó que la tecnología a utilizar para llevar a cabo este modelo de aplicación es Backbone.js. Ahora vamos a marcar las pautas de diseño para que la aplicación sea lo más robusta posible.

En primer lugar, dado que Backbone nos proporciona la definición de vistas y de modelos, vamos a crear un par vista-modelo que implementen la funcionalidad común a todas las vistas de nuestra aplicación. Denominaremos a este par de elementos vista base y modelo base. Estos elementos serán extendidos por el resto de vistas y modelos de la aplicación.

Nuestra vista base implementará principalmente dos funciones de la View de Backbone, que son:

- **initialize:** Función que inicializa la vista, en ella se define la información necesaria para la vista. En nuestro caso, al ejecutarse esta función, se combinarán los eventos de esta vista común con los eventos de cualquier vista que extienda de ella. Además se añade la función *aferInitialize*, que podrá ser ejecutada por las vistas justo después de esta función.
- **render:** Esta función es la encargada de "pintar" la vista, es decir, se encargará de integrar los datos del modelo en la plantilla de la vista. En la vista base de la aplicación, se han definido un par de funciones que se ejecutan antes y después de la renderización, para dar la posibilidad de ejecutar alguna tarea en estos momentos, en cualquier vista. Estas funciones de llaman *beforeRender* y *afterRender* respectivamente.

Para aportar más información se adjunta la siguiente imagen con los diagramas de flujo que seguirán estas funciones en las distintas vistas de nuestra aplicación.

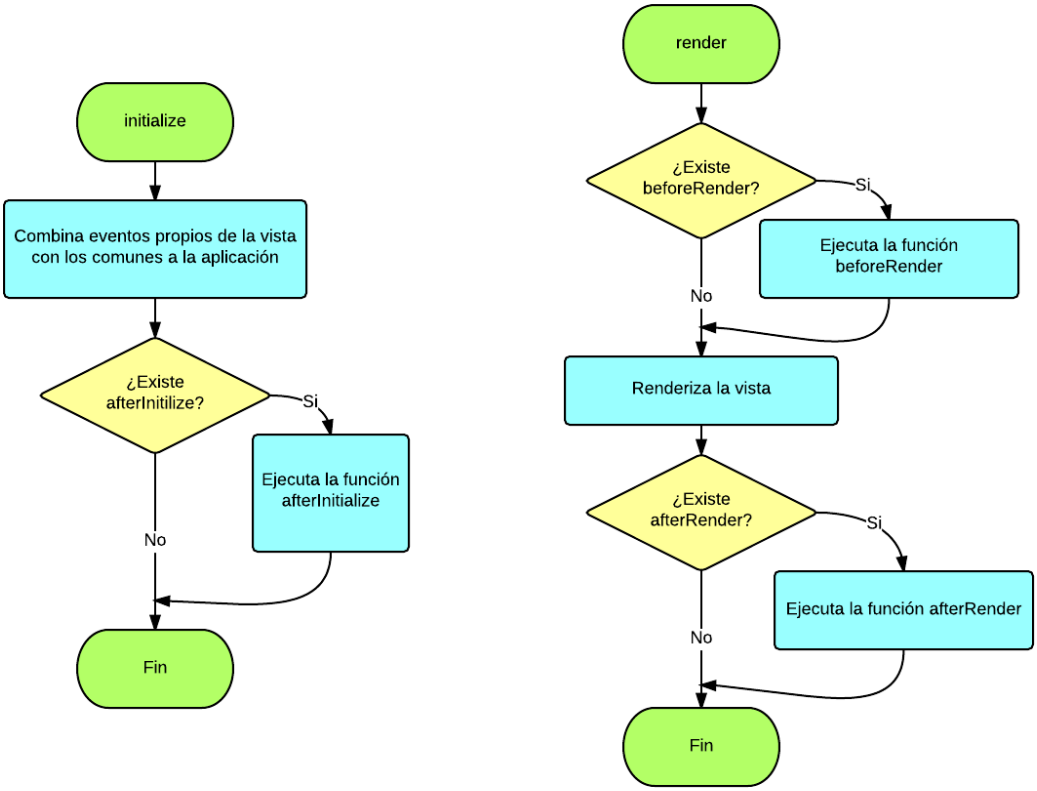

Ilustración 9: Funciones de la vista base de la aplicación

Siempre que nuestra vista necesite información del servidor (como la obtención de datos del usuario), se establecerá la función afterInitialize, como el lugar donde establecer la comunicación con el back-end por medio de peticiones AJAX. De esta forma podremos cargar los datos en el modelo antes de que se ejecute la función *render*.

El modelo base de nuestra aplicación de videoconferencia será una réplica del modelo de Backbone. El sentido de generar un modelo base, en lugar de usar el modelo del framework, es para dejar la aplicación preparada en caso de que en un futuro, se quiera añadir información al modelo, común a todas las vistas de la aplicación.

Por otro lado, se creará un objeto en global en Javascript para la aplicación, es decir, un objeto con información de la aplicación, al que se podrá acceder desde cualquier sección la página. Este objeto se definirá como *$V* y contendrá información de la aplicación, tal como, el nombre, el título mostrado en el navegador, las rutas de los ficheros estáticos, etc., y además, la vistas, los modelos y clase auxiliares.

```
$V = {
  appName: 'Videoconferencia',
  appTitle: 'Videoconferencia sobre HTML5',
  version: '1.0.0',
  templatesReady: false,
  jsLibs: {
    route: 'libs/',
    versions: {
      jquery: '1.10.2',
      underscore: '1.6.0',
      backbone: '1.1.2',
      requireText: '2.0.12',
      bootstrap: '3.2.0',
      socketio: '1.1.0'
    }
  },
  templates: {
    route: '../tmpl/'
  }
  views: {}
};
```

Ilustración 10: Información contenida en $V

Por último, todas las plantillas que se utilizan para generar las vistas de la aplicación se definirán como script de tipo *text/template*. Todas estas plantillas o templates se añadirán al final de la página HTML de la aplicación para que sean accesibles desde cualquier estado de la aplicación, y así, poder ser renderizados cuando proceda.

Vamos a definir dos contenedores dentro de la etiqueta <body> de nuestra aplicación, el primero de ellos tendrá el id "contenedor", y será donde se aloje el código HTML de la vista; el segundo contenedor, contendrá todos los templates de las vistas de la aplicación. En la siguiente figura se muestra esta estructura de contenedores que posee la aplicación de videoconferencia.

```
<!DOCTYPE html>
<html lang="es">
  <head>…</head>
  <body>
    <div id="contenedor" class="container-fluid">…</div>
    <div id="templates-area">
      <script type="text/template" id="tmpl_main">
        <div id="page-content" class="container"></div>
      </script>
      <script type="text/template" id="tmpl_home">…</script>
      <script type="text/template" id="tmpl_navBar">…</script>
      <script type="text/template" id="tmpl_videochat">…</script>
      <script type="text/template" id="tmpl_contacts">…</script>
      <script type="text/template" id="tmpl_login">…</script>
      <script type="text/template" id="tmpl_login_home">…</script>
      <script type="text/template" id="tmpl_login_signIn">…</script>
      <script type="text/template" id="tmpl_login_signUp">…</script>
      <script type="text/template" id="tmpl_signup">…</script>
      <script type="text/template" id="signupForm">…</script>
    </div>
  </body>
</html>
```

Ilustración 11: Contenedor de templates

4.3 Implementación

En este apartado se enumerarán algunos de los aspectos más importantes que se han tenido en cuenta a la hora de desarrollar este trabajo de fin de grado.

Inicialización de la aplicación

Cuando se accede a la aplicación de videoconferencia desde un navegador, llega una petición **GET** al servidor para la ruta **"/"** (más adelante veremos los tipos de peticiones que nuestro servidor ha de resolver), y éste devuelve una respuesta con la página index.html de la aplicación.

Esta página HTML es muy sencilla, pues en el cuerpo, cuenta solamente con los dos contenedores vistos en la anterior sección de diseño de la aplicación; y en la cabecera, incluye, las hojas de estilos, y el script de la librería require.js, al que se le indica como parámetro, un fichero Javascript donde se inicializa la aplicación (vc.main.js).

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>{{ $V.appTitle }}</title>
    <meta name="csrf-token" content="{{ csrfToken }}">

    <link rel="stylesheet" href="css/bootstrap.css" />
    <link rel="stylesheet" href="css/main.css" />

    <script data-main="js/vc.main" src="js/libs/require-2.1.14.js"></script>
  </head>
  <body>
    <div id="contenedor" class="container-fluid"></div>
    <div id="templates-area"></div>
  </body>
</html>
```

Ilustración 12: index.html

En este fichero Javascript se define el objeto de la aplicación **$V**, así como el árbol de dependencias de la aplicación, para que la librería require.js se encargue de importarlas correctamente al head de nuestra página HTML.

Este fichero Javascript también es el encargado cargar todas y cada una de las plantillas de las vistas e inyectarlos en el contenedor de templates. Una vez que se se han cargado tanto las dependencias de los ficheros Javascript, como los templates en el HTML, se crea la primera vista de la aplicación, y a partir de este momento la aplicación comienza a funcionar.

Gestión de vistas

En esta sección veremos cómo la aplicación es capaz de mostrar las distintas vistas del sitio web.

Tal y como se vio en los requisitos, la aplicación ha de saber si el usuario ha iniciado sesión en la aplicación o no, y en base a esto, mostrar unas vistas u otras. En este caso vamos a ver qué sucede cuando un usuario accede a la aplicación desde un navegador.

Una vez la aplicación se inicializa se crea la primera vista denominada MainView. Esta vista es la encargada de saber si el usuario ya tiene sesión o no para mostrar la pantalla de bienvenida o la Home del usuario.

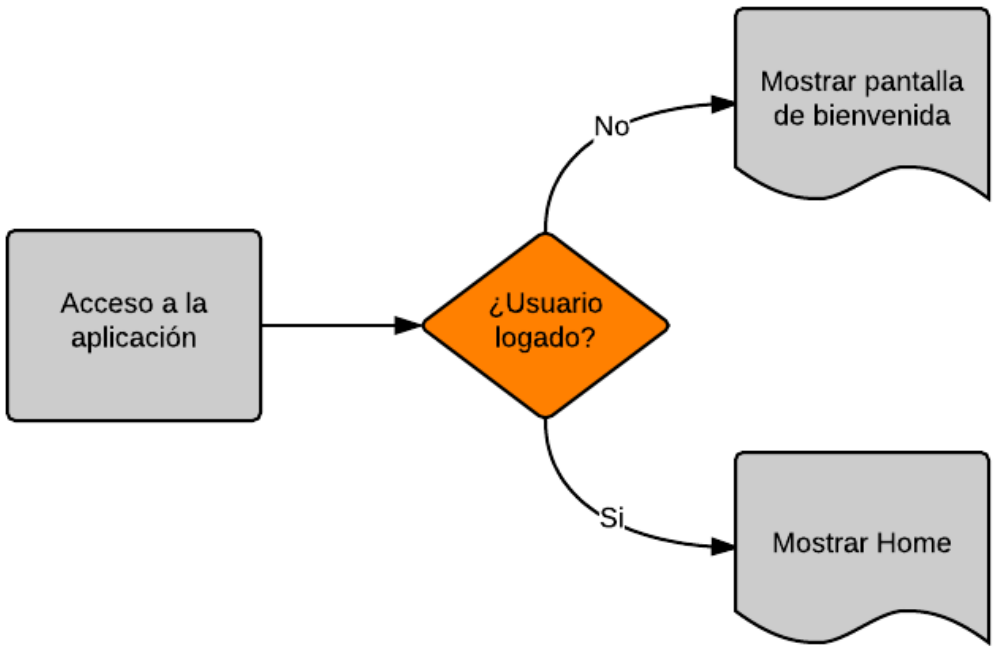

Ilustración 13: Acceso a la aplicación

Para saber si el usuario está logado, se hará uso de la función **afterInitialize** para establecer comunicación con el servidor con el fin de saber si el usuario tiene sesión o no. Después se utilizará la función **afterRender** para crear la vista correspondiente.

Aunque no se ha mencionado, cada una de las vistas definidas ha de tener asociado un template que indica la plantilla a renderizar y un contenedor para la vista.

En el caso de que el cambio de vista sea producido por una navegación, se utilizará el script vc.router.js, que extiende de la clase Router de Backbone, y gestionará la navegación creando la vista correspondiente en cada momento.

Establecimiento de sesiones

Acabamos de ver que cuando se accede a la aplicación, se comprueba si el usuario está logado. En esta sección, se va a explicar cómo se establecen las sesiones y cómo se sabe si el usuario ya está logado o no.

La manera en que se establecen sesiones en la aplicación es a través de cookies almacenadas en el navegador del usuario.

Cada vez que un usuario inicia sesión en la aplicación, se almacenan dos cookies en su navegador, que contienen su nombre de usuario y contraseña encriptados respectivamente. Estas cookies tienen una caducidad de un día, por lo que, si un usuario inicia sesión en la aplicación y sale de ella sin cerrar sesión, la próxima vez que acceda en un plazo de tiempo inferior a un día, se le mostrará directamente la Home.

Para saber si un usuario está logado o no en la aplicación se hará una consulta a la base de datos con la información de las cookies, y si el usuario se encuentra en la base de datos, podremos afirmar que ya había iniciado sesión con anterioridad. En caso de que el usuario no hubiera iniciado sesión, como no va a tener las cookies almacenadas en el navegador, cuando se haga la consulta a la base de datos, no se obtendrán resultados.

Cuando un usuario elige la opción “Desconectar” para salir de la aplicación, se eliminan las cookies del navegador con su usuario y contraseña.

Servidor de la aplicación

La implementación del servidor ha ido evolucionando al ritmo de la aplicación. En una primera fase de desarrollo, se implementó un servidor de contenidos estáticos sin la ayuda de ningún modulo de Node.js, con el objetivo de dar los primeros pasos en Node.js (25).

Después se tomó la decisión de usar Express.js, ya que nos permite crear y configurar un servidor de manera bastante sencilla. Aunque finalmente se optó por usar el módulo express-io, que además, incluye el manejo de sockets para las comunicaciones en tiempo real que requiere la aplicación de videoconferencia.

En el servidor de la aplicación que nos ocupa se ha configurado, entre otros, los siguientes parámetros:

- Se establece el puerto 8001 para escuchar las peticiones del cliente.
- Se configura el servidor para que sea capaz de almacenar cookies en el navegador.
- Se define la ruta donde se encuentran los archivos estáticos de la aplicación, es decir, la ruta donde se encuentran todos los archivos que conforman el cliente de la aplicación.
- Además se define la ruta dónde se encuentra el archivo index.html de la aplicación, que se tuvo que separar del resto de archivos estáticos, para que la aplicación no lo sirviera automáticamente al acceder desde el navegador. Esta modificación fue necesaria para añadir un token CSRF a nuestra aplicación.
- Por último se definen las rutas a las que se harán las peticiones desde el cliente para obtener los datos y cargarlos en el modelo de la vista.

Cada vez que se arranca el servidor de la aplicación se realizan las siguientes tareas:

1) Se establece conexión con la base de datos de usuarios.
2) Se aplica la configuración al servidor.
3) El servidor permanece escuchando las peticiones del cliente.

Base de datos

Para establecer la comunicación con la base de datos, el servidor hace uso del framework Mongooose.js, que nos permite crear un esquema para almacenar los datos de los usuarios.

El esquema de usuarios de la aplicación almacenará objetos que tengan los siguientes datos:

- **username:** String que indica el nombre de usuario, ha de ser único.
- **email:** String para indicar el email del usuario.
- **hash:** Contraseña del usuario después de aplicarle una encriptación.
- **state:** Número entero entre 0 y 2 que indica el estado del usuario: 0 indica que el usuario está disponible, 1 que no está disponible y 2 que está ocupado.

Además de estos campos, se añade un campo **_id** automáticamente a cada registro insertado en la base de datos.

Se ha añadido una mejora a este esquema, para que antes de almacenar la información de un usuario, compruebe si ya existe ese username, y en tal caso, no se pueda almacenar el usuario, y así, asegurar que el nombre de usuario es único.

Gestión de usuarios

La gestión de usuarios se hace a través de consultas a la base de datos de la aplicación. Estas consultas se realizarán desde el servidor cada vez que se soliciten desde el cliente.

En el servidor de la aplicación están definidas una serie de rutas, a las que el cliente realizará peticiones AJAX, para realizar la gestión de usuarios.

- En el caso de que el cliente quiera saber si el usuario está logado en la aplicación, éste enviará una petición de tipo **GET** a la url "**api/auth",** y recibirá como respuesta el estado del usuario. La lógica para determinar si el usuario está logado ya ha sido explicada anteriormente.

- Cuando un usuario quiera cerrar sesión, habrá que enviar una petición de tipo **POST** a la url **"api/auth/logout"** para que se inicie el proceso para cerrar sesión descrito anteriormente.

- Para iniciar sesión en la aplicación, la petición se realizará a la url "**api/auth"**, pero en este caso, será de tipo **POST**, indicando el nombre de usuario y contraseña. La lógica que se ejecutará en el servidor seguirá el siguiente esquema:

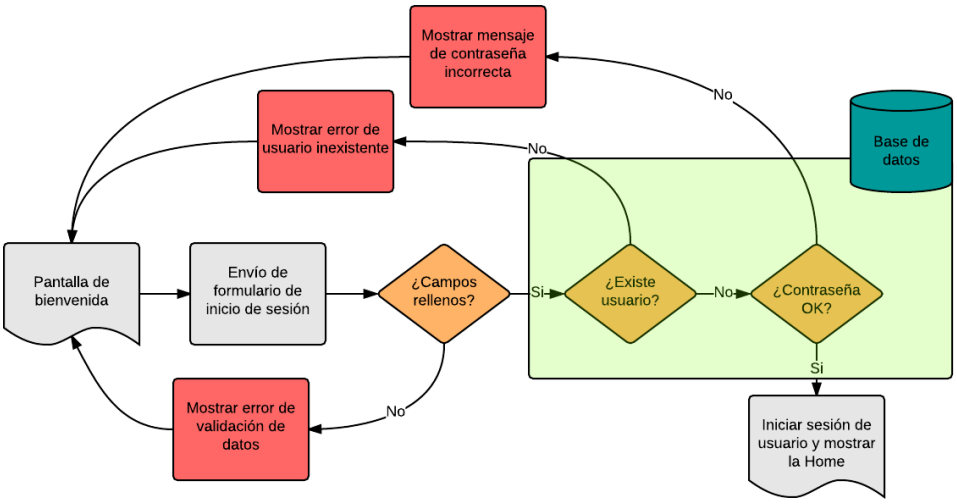

Ilustración 14: Flujo de inicio de sesión

- Para dar de alta a nuevos usuarios en la aplicación, las peticiones se realizarán a la ruta **"api/auth/signup"** y serán de tipo **POST**, indicando los datos de registro del usuario. En este caso se aplicará la siguiente lógica:

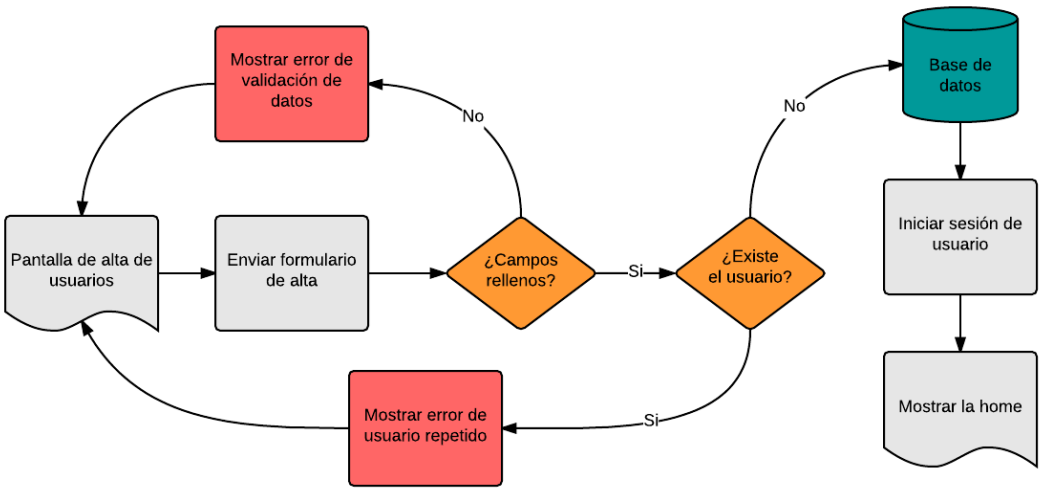

Ilustración 15: Flujo de alta de usuario

En este caso no se enviará la petición al servidor si alguno de los campos no está relleno. En caso de que el nombre de usuario ya exista en la base de datos, se informará al cliente del error en la respuesta, y en caso contrario, se añadirá un nuevo registro a la base de datos y se iniciará la sesión de usuario.

Seguridad

Uno de los requisitos que se pedía para la implementación de este proyecto es el de dotar a la aplicación de seguridad. Para ello se han tomado dos medidas fundamentalmente.

En primer lugar, la aplicación tiene implementado un servidor con seguridad HTTPS. Para hacer esto posible hubo que crear unos certificados de seguridad por medio de OpenSSL.

Por otro lado, cuando se accede a nuestra aplicación, se inyecta un token CSRF en la cabecera HTML. Este token deberá estar incluido en la cabecera de las peticiones que se envíen al servidor para evitar ataques por CSRF.

De este modo todas las peticiones que no incluyan este parámetro en la cabecera serán rechazadas por el servidor.

Además de estos procedimientos para añadir seguridad a la aplicación, se ha

tenido bastante cuidado con el manejo de datos sensibles del usuario, con el objetivo de cumplir con la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) (26):

- Al añadir a un usuario a la base de datos se realiza una encriptación de la contraseña. Para realizar esta encriptación se utiliza el modulo bcrypt-nodejs de Node.
- Para determinar si el usuario está logado o no, en lugar de usar el nombre de usuario y la contraseña, se usan los códigos generados en las cookies a partir de estos datos.
- En las respuestas que se reciben del servidor con datos del usuario, se omiten los datos sensibles para evitar que aparezcan estos datos al utilizar las herramientas de desarrollador que incluyen los navegadores.

Home del usuario

Cada vez que un usuario accede a la home, internamente en la aplicación, se envía una petición **GET** a la url **"api/home"**. La respuesta de esta aplicación será un objeto JSON que contendrá los siguientes campos:

contacts: Una lista con todos los usuarios de la aplicación en la que cada elemento será un objeto con los siguientes campos

- **id:** Indice del usuario en la base de datos (único por usuario).
- **name:** Nombre del usuario.
- **state:** Entero que indica la disponibilidad del usuario.

user: Un objeto con todos los datos del usuario.

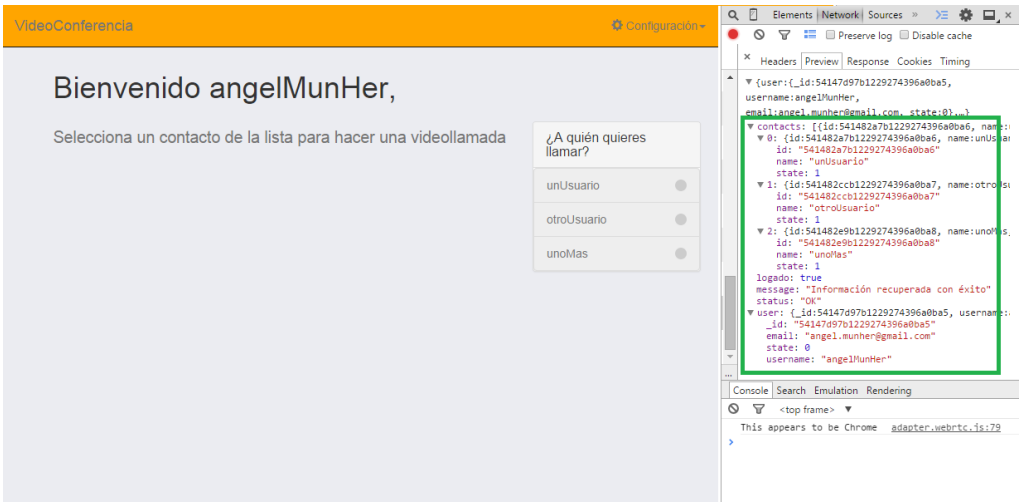

Ilustración 16: Captura de datos de la home

Tal y como se muestra en la anterior figura, los datos obtenidos en la respuesta a esta petición se añaden al modelo para que sean mostrados en la vista, añadiendo el nombre del usuario en el contenedor principal y el listado de usuarios en el contenedor secundario.

Captura de audio y vídeo

Para la captura de audio y vídeo, así como para la implementación del módulo de videoconferencia, se ha hecho uso del API de la tecnología WebRTC.

En particular para esta cuestión se utilizará la función *getUserMedia()* de WebRTC, que es la encargada de adquirir los flujos de datos emitidos por los dispositivos multimedia (micrófono y webcam) del usuario.

A esta función hay que pasarle tres parámetros:

- **constraints:** Los flujos de datos que se quieren capturar. En nuestro caso audio y vídeo
- **successCallback:** Función que se ejecuta una vez que se obtienen los flujos de datos multimedia.
- **errorCallback:** Función que se ejecuta en caso de que se produzca un error al obtener los flujos multimedia.

Una vez que el navegador obtiene los flujos multimedia, los encapsula en una url, y esta url es la que se añade a la etiqueta video de HTML5 que representa al vídeo local, y después, se inicia la reproducción del vídeo.

Para detener la captura se proporciona al usuario un botón que se encarga de parar el vídeo y eliminar el flujo de datos multimedia.

Módulo de videoconferencia

La implementación de este modulo no se ha llegado a finalizar. Por falta de tiempo y de conocimiento de trabajo con sockets.

La idea perseguida para establecer la comunicación entre los usuarios es que un usuario seleccione un contacto de la lista de usuarios, y al hacer click en el usuario seleccionado, comience el proceso de videollamada.

Para hacer esto posible, cada elemento de la lista de usuario incluye un atributo *data-user* con el índice asociado a cada usuario en la base de datos. Con esta información podremos indicar al servidor el usuario final de la comunicación.

El proceso a implementar debería ser el siguiente:

1) El usuario selecciona un contacto con estado disponible de la lista de contactos. Con este proceso se inicia la captura local de audio y video.
2) Se crea una sala de chat y se envía una oferta SDP al usuario remoto. Para ello será necesario informar al servidor del id de usuario para obtener su dirección IP y poder establecer comunicación utilizando sockets.
3) Una vez que el usuario remoto recibe la oferta SDP, se le mostrará un aviso informando de que un usuario quiere establecer una videollamada con él.
4) Si este usuario acepta la videollamada se creará una respuesta SDP aceptando la oferta y se iniciará la comunicación. Además iniciará la captura de flujos multimedia.
5) Cuando llegue la respuesta afirmativa para el establecimiento de la sesión multimedia se añadirá la url del flujo remoto al video mostrado por pantalla.

Para la implementación de este módulo se siguió un ejemplo de desarrollo con WebRTC que aparece en la página web html5rocks (27).

Capítulo 5: Gestión del proyecto

En este capítulo se detallará la planificación que se ha seguido para el desarrollo de este trabajo de fin de grado y se estimará el coste que supone su desarrollo a modo de presupuesto.

5.1 Planificación

A continuación se muestra un desglose de las fases de este proyecto de fin de grado y el tiempo dedicado para cada una de ellas:

Fase de desarrollo	Fecha de inicio	Fecha de fin	Duración en días
Planificación	26/05/2014	03/06/2014	7
Estudio del estado del arte	04/06/2014	10/06/2014	7
Formación inicial	11/06/2014	01/07/2014	
Formación inicial de Node.js	11/06/2014	24/06/2014	14
Formación de MongoDB	25/06/2014	01/07/2014	7
Análisis de requisitos	02/07/2014	15/07/2014	
Definición de requisitos	02/07/2014	08/07/2014	7
Definición de casos de uso	09/07/2014	15/07/2014	7
Diseño de la arquitectura	16/07/2014	05/08/2014	21
Implementación y pruebas	06/08/2014	04/09/2014	30
Memoria y documentación	05/09/2014	24/09/2014	20
			120

Tabla 34: Planificación del proyecto

Si establecemos que por motivos de trabajo, sólo se ha podido dedicar al desarrollo de este proyecto una media de tres horas diarias, la suma de horas invertidas en la realización del trabajo de fin de grado serían 360 horas. Este tiempo se ajusta los 12 créditos ECTS que suponen la realización de trabajo de fin de grado.

5.2 Presupuesto

Para realizar el cálculo del presupuesto se tendrán en cuenta los siguientes factores:

- Horas de trabajo desempeñadas.
- Elementos de hardware.
- Licencias software.

En cuanto a las licencias de software, en el caso de este trabajo de fin de grado, el coste sería 0 ya que todo el software utilizado es de código libre.

En el caso del hardware, para estimar los costes, hay que tener en cuenta que, para el desarrollo de nuestra aplicación, se ha necesitado el empleo del ordenador personal y un ordenador del departamento de telemática para el alojamiento de la web y la base de datos.

Dado que no disponemos información sobre los costes del equipo donde está alojada nuestra aplicación, vamos a aportar una solución alternativa para alojar la web, y así, poder ofrecer la información de los costes. Esta solución alternativa consiste en utilizar un alojamiento web que soporte Node.js y bases de datos MongoDB.

Se han encontrado distintas plataformas que ofrecen alojamiento web para este tipo de aplicaciones. Todos estos sitios ofrecen alojamiento web en distintos paquetes en función de del trafico de usuarios y la cantidad de datos que maneje la aplicación. Y todos ellos incluyen un paquete gratis que permiten alojar una aplicación web con bajo número de visitas y bajo almacenamiento de datos. Entre estos sitios se encuentran OpenShift y Nodejitsu.

Por tanto, vamos a asumir que no tenemos coste para el alojamiento del servidor de la aplicación, ya que los paquetes gratuitos de los sitios de alojamiento web cubren nuestras necesidades actuales. Conforme vaya creciendo la afluencia de usuarios en nuestra aplicación, habrá que tomar la decisión de adquirir el paquete de alojamiento que más se ajuste a nuestras necesidades.

Por otro lado, para la realización de este proyecto se ha utilizado un ordenador portátil, modelo Samsung NP300V5A-S03ESPC, con procesador Intel Core i7, 4 GB de RAM y sistema operativo Windows 7 Home Premium, valorado en unos 700 €.

Por último, tal y cómo hemos visto en el apartado anterior, se han dedicado 360 horas al desarrollo de este trabajo de fin de grado. Ahora necesitamos hacer un cálculo para saber el coste que ha supuesto la realización del trabajo.

Consultando distintas páginas web, se ha encontrado que el precio por hora que se utiliza para el desarrollo de aplicaciones web varía entre 20 y 50 euros en función del perfil del desarrollador. Dado que mi perfil de desarrollador no es muy alto, se va a establecer un coste de desarrollo de 25€ por hora. Por lo que, el coste del desarrollo de este proyecto asciende a la cantidad de 9000 euros (360 horas x 25 €/hora).

Por tanto, el presupuesto total de este trabajo de fin de grado es de **9700 euros**. A esta cantidad habría que sumarle el coste mensual del alojamiento web en caso de elegir un paquete premium.

Capítulo 6: Conclusiones y trabajos futuros

En este capítulo hablaremos sobre las conclusiones obtenidas en base a los objetivos perseguidos durante la realización del trabajo de fin de grado. Estas conclusiones también reflejarán la consecución de los requisitos de la aplicación desarrollada.

Por otro lado se aportaran una serie de mejoras a introducir en el desarrollo actual, para conseguir una aplicación más completa.

6.1 Conclusiones

Dado que el objetivo de este trabajo de fin de grado es el de desarrollar una aplicación capaz de establecer sesiones de videoconferencia entre usuarios, no podemos decir que se hayan cumplido todos los objetivos, porque aunque se ha desarrollado y creado la aplicación, no se ha logrado su funcionamiento al 100%. Aunque sí que se han conseguido una serie de objetivos parciales relacionados con el desarrollo de esta aplicación.

En primer lugar se ha conseguido desarrollar una aplicación web desde cero. Y para ello se ha hecho uso de tecnología de vanguardia en el desarrollo de aplicaciones web.

Para poder desarrollar esta aplicación se ha tenido que hacer un trabajo de investigación del estado actual de la tecnología. Así como una gran labor de aprendizaje de nuevas tecnologías de desarrollo.

Se ha tenido especial cuidado a la hora de definir la arquitectura de la aplicación con el motivo de poder permitir la fácil inclusión de nuevos desarrollos que permitan el crecimiento de la aplicación. Además se ha intentado definir la arquitectura de la aplicación de manera que pueda ser utilizada para el desarrollo de nuevas aplicaciones web.

Por otro lado, teniendo en cuenta los distintos requisitos y casos de uso definidos para la aplicación en la fase de análisis, se han cumplido prácticamente todos los objetivos propuestos, con la salvedad del establecimiento de la sesión de videoconferencia.

Otro de los puntos en los que quiero hacer hincapié es la seguridad que se ha implementado en esta aplicación, pues el desarrollo siempre ha estado centrado en la protección de la privacidad y los datos sensibles del usuario para dar conformidad con

la LOPD.

Por último quiero destacar que todo el trabajo realizado me ha servido, tanto para mi desarrollo personal, como profesional, ya que toda la tecnología utilizada durante el desarrollo, es tecnología que puedo utilizar en mi futuro profesional.

Por todo lo expuesto anteriormente, podemos considerar que, aunque la consecución de objetivos no ha sido del todo satisfactoria, los resultados obtenidos han sido bastante buenos.

6.2 Trabajos futuros

El primer trabajo a realizar es hacer funcionar la comunicación entre los usuarios de la aplicación, para que puedan establecerse las videoconferencias. Para ello será necesario adquirir conocimientos sobre el uso se sockets.

Una vez conseguido esto, podremos incluir los siguientes desarrollos para mejorar la calidad de la aplicación desarrollada:

- Dar un enfoque de red social a la aplicación, de manera que, en lugar de mostrar el listado de todos los usuarios de la aplicación, permitir al usuario definir un listado de usuarios con sus amigos.

- Además de esto, también sería importante dar la posibilidad al usuario de eliminar su cuenta, pues este desarrollo no se ha tenido en cuenta en la realización de este proyecto.

- Otra de las mejoras a realizar es facilitar la autenticación en la aplicación utilizando redes sociales, es decir, permitir que los usuarios se puedan dar de alta ligando su cuenta a su cuenta de Facebook o de Twitter por ejemplo.

- Un desarrollo importante que se debe incluir en la aplicación es el de introducir publicidad para obtener beneficios. Y de esta manera intentar cubrir los costes vistos en el capítulo de gestión del proyecto.

- Mejorar la accesibilidad de la aplicación, pues en este desarrollo no se ha tenido en cuenta temas relacionados con la accesibilidad.

Glosario

AJAX: Técnica de desarrollo web para crear aplicaciones interactivas. Permite el envío de datos entre cliente y servidor de forma asíncrona de manera que no influye al funcionamiento de la aplicación.

API: Conjunto de funciones o procedimientos que ofrece cierta biblioteca para ser utilizados como una capa de abstracción.

ASP: Tecnología de Microsoft para desarrollar la parte servidor de las aplicaciones web.

Back-end: Se refiere a la parte servidor de la aplicación. Es la encargada de manejar las peticiones del cliente y establecer comunicación con la base de datos para almacenar la información.

Blog: Sitio web en el que uno o varios autores publican artículos periódicamente.

Códec: Especificación desarrollada en software, hardware o una combinación de los dos para transformar un archivo con un flujo de datos o una señal. Los códecs pueden codificar estos archivos para la transmisión/almacenaje/cifrado y descodificarlos para la reproducción.

Cookie: Información de un sitio web que se queda almacenada en el navegador, generalmente almacenada para guardar datos de sesión y preferencias.

CSRF: Cross-site request forgery. Es un tipo de ataque que utiliza a un usuario validado para, a través de éste, introducir solicitudes "válidas" que modifiquen el comportamiento de la aplicación a favor del atacante.

CSS/CSS3: Hoja de estilos en cascada. Lenguaje usado para la definición de estilos aplicados a documentos HTML o XML.

DOM: Estructura de objetos que genera el navegador cuando se carga un documento HTML. Puede modificarse mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página.

Flash: Aplicación de creación y manipulación de gráficos vectoriales.

Framework: Conjunto de herramientas que ofrecen una funcionalidad aportando una capa de abstracción.

Front-end: Se refiere a la parte cliente de las aplicaciones web. Se encarga de ofrecer los contenidos de manera estructurada y de interactuar con el usuario.

G.711: Estándar de la ITU-T para la codificación de audio.

H.264: Estándar de compresión de vídeo desarrollado por la ITU-T Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG).

Hipertexto: Herramienta de software que permite crear, agregar, enlazar y compartir información por medio de enlaces asociativos.

HTML/HTML5: HyperText Markup Language. Lenguaje de programación de páginas web

HTTP: Protocolo de transferencia de hipertexto. Define la sintaxis y la semántica que utilizan los elementos del software de la arquitectura web para comunicarse.

HTTPS: Protocolo basado en HTTP destinado para la transferencia segura de datos de hipertexto.

IETF: Organización internacional de normalización cuyo objetivo es velar por que la arquitectura y los protocolos que conforman Internet funcionen correctamente.

Javascript: Lenguaje de programación utilizado principalmente en el desarrollo de aplicaciones web.

JSON: Formato ligero de intercambio de datos constituido por dos estructuras: una colección de pares nombre/valor, y una lista ordenada de valores.

Mixin: Clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero que no está ideada para ser autónoma.

MPEG: Grupo de trabajo de expertos formado por ISO Y IEC para establecer estándares para el audio y la transmisión de vídeo.

NExT: Compañía estadounidense de informática, formada por Steve Jobs, creadora del NExT Computer, que fue usado por Tim Berners-Lee como primer servidor web del mundo.

NPM: Gestor de paquetes de Node.js. Este módulo se instala automáticamente con Node.js y permite la gestión de dependencias del proyecto por línea de comandos.

Opus: Códec de audio desarrollado por la IETF pensado para aplicaciones que usen sonido a tiempo real en Internet.

PHP: Lenguaje de programación de la parte servidor de aplicaciones web.

Plugin: Complemento que añade cierta funcionalidad al software.

Responsive design: Filosofía de diseño y desarrollo de aplicaciones web cuyo objetivo es adaptar el contenido de las páginas web al dispositivo en el que se visualizan.

RTP: Real-time Transport Protocol. Protocolo utilizado en la transmisión de información en tiempo real.

RTCP: Real Time Control Protocol. Protocolo de comunicación que proporciona información de control asociado a un flujo de datos para una aplicación multimedia.

SIP: Session Initiation Protocol. Protocolo de señalización utilizado en voz sobre IP.

SPI: Single Page Interface. Modelo de sitio o aplicación web que consta de una sola página cuya transición entre vistas se hace a través de eventos.

SSH: Protocolo que permite acceder a máquinas remotas a través de una red.

Socket: Mecanismo que permite la conexión de dos procesos. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Template: Script con código HTML que conforma la plantilla donde se mostrarán los datos de una vista.

VoIP: Voz sobre IP. Grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP.

VP8: Códec de vídeo desarrollado por Google utilizado en el API WebRTC.

W3C: World Wide Web Consortium. Consorcio internacional que produce recomendaciones para la Web.

WAI-ARIA: Iniciativa del W3C que define cómo hacer accesibles contenidos y aplicaciones web.

Webmaster: Persona que realiza el mantenimiento de páginas web. En la Web 1.0 es el encargado de publicar y actualizar los contenidos de los sitios web.

Wiki: Sitio web cuyas páginas pueden ser editadas directamente desde el navegador, donde los usuarios crean, modifican o eliminan contenidos.

XML: eXtensible Markup Language. Lenguaje de marcas utilizado para almacenar datos de forma legible. Lenguaje útil para realizar la comunicación entre aplicaciones.

Bibliografía

1. W3C. *Tim Berners-Lee.* [En línea] [Citado el: 18 de 09 de 2014.] http://www.w3.org/People/Berners-Lee/.

2. **De Luca, Damián.** RedUSERS. *¿Por qué HTML5 está reemplazando a Flash? La verdadera historia.* [En línea] 06 de 01 de 2012. [Citado el: 09 de 18 de 2014.] http://www.redusers.com/noticias/porque-html5-esta-reemplazando-a-flash/.

3. **Arranz Santamaría, Jose María.** itsnat. *Manifiesto por una Única Página Web (Single Page Interface).* [En línea] 15 de 07 de 2011. [Citado el: 19 de 09 de 2014.] http://itsnat.sourceforge.net/php/spim/spi_manifesto_es.php.

4. Node.js. [En línea] [Citado el: 22 de 09 de 2014.] http://nodejs.org/.

5. **Harrel, Jeff.** PayPal Engineering. *Node.js at PayPal.* [En línea] 22 de 11 de 2013. [Citado el: 18 de 09 de 2014.] https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/.

6. **Gutiérrez, Pedro.** GENBETA:dev. *Responsive Design: introducción.* [En línea] 15 de 11 de 2012. [Citado el: 21 de 09 de 2014.] http://www.genbetadev.com/desarrollo-web/responsive-design-introduccion.

7. Portal de Administración electrónica. *Normas Accesibilidad.* [En línea] [Citado el: 18 de 09 de 2014.] http://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Accesibilidad/pae_normativa/pae_eInclusion_Normas_Accesibilidad.html#.VB9SOvl_tQJ.

8. **Fernández Rivera, Javier.** WAI-ARIA, una aproximación. *WAI-ARIA, una aproximación.* [En línea] 04 de 02 de 2009. [Citado el: 18 de 09 de 2014.] http://www.nosolousabilidad.com/articulos/wai_aria.htm.

9. **Adam Bergkvist, Ericsson, y otros, [ed.].** W3C Editor's Draft. *WebRTC 1.0: Real-time Communication Between Browsers.* 01 de julio de 2014.

10. *WebRTC: comunicaciones en tiempo real en el navegador Web.* **Millán Tejedor, Ramón Jesús.** 173, s.l. : GM2 Publicaciones Técnicas, 10 de 03 de 2014, Conectrónica.

11. **Paoli, Jean.** MSDN Blogs. *Customizable, Ubiquitous Real Time Communication over the Web (CU-RTC-Web).* [En línea] 06 de 08 de 2012. [Citado el: 22 de 09 de 2014.]

http://blogs.msdn.com/b/interoperability/archive/2012/07/28/customizable-ubiquitous-real-time-communication-over-the-web-cu-rtc-web.aspx.

12. Backbone.js. [En línea] [Citado el: 22 de 09 de 2014.] http://backbonejs.org/.

13. Angular.js. [En línea] [Citado el: 22 de 09 de 2014.] https://angularjs.org/.

14. Polymer. [En línea] [Citado el: 22 de 09 de 2014.] https://www.polymer-project.org/.

15. jQuery. [En línea] [Citado el: 22 de 09 de 2014.] http://jquery.com/.

16. Underscore.js. [En línea] [Citado el: 22 de 09 de 2014.] http://underscorejs.org/.

17. Require.js. [En línea] [Citado el: 22 de 09 de 2014.] http://requirejs.org/.

18. Socket.io. [En línea] [Citado el: 22 de 09 de 2014.] http://socket.io/.

19. Bootstrap. [En línea] [Citado el: 22 de 09 de 2014.] http://getbootstrap.com/.

20. MongoDB. [En línea] [Citado el: 22 de 09 de 2014.] http://www.mongodb.org/.

21. Express-io. [En línea] [Citado el: 22 de 09 de 2014.] http://express-io.org/.

22. bcrypt-nodejs. [En línea] [Citado el: 22 de 09 de 2014.] https://github.com/shaneGirish/bcrypt-nodejs.

23. Mongoose.js. [En línea] [Citado el: 22 de 09 de 2014.] http://mongoosejs.com/.

24. **Maulini Rubiera, Mauro.** Desarrollo y Seguridad de Aplicaciones Web y Móviles. *¿Que es el Cross Site Request Forgery (CSRF)?* [En línea] 13 de 06 de 2012. [Citado el: 22 de 09 de 2014.] http://tecnologiasweb.blogspot.com.es/2010/12/que-es-el-cross-site-request-forgery.html.

25. **Kiessling, Manuel.** *The Node Beginner Book.* s.l. : Leanpub, 2012. ISBN: 978-1-4716-2844-3.

26. Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

27. **Dutton, Sam.** HTML5 Rocks. *Getting Started with WebRTC.* [En línea] 23 de 07 de 2012. [Citado el: 22 de 09 de 2014.] http://www.html5rocks.com/en/tutorials/webrtc/basics/.

Anexo I: Instalación y despliegue de la aplicación

Requisitos:

Será necesario tener instalados en nuestra máquina:

1) **Node.js**: al menos la versión v0.10.22 o superior.
2) **MongoDB:** al menos la versión 2.6.4 o superior.
3) **Git:** necesario para descargar el código de la aplicación.

Además será necesario disponer de los certificados SSL para el servidor HTTPS. Estos certificados se pueden generar de manera sencilla usando OpenSSL.

Instalación:

El código de la aplicación se encuentra disponible en github:

https://github.com/mimbrin/Videoconferencia

Una vez descargado el código, habrá que crear una nueva carpeta con el nombre **ssl** en la raíz del proyecto, y en ella añadir los certificados de seguridad. El nombre de los certificados ha de ser:

- **server.key:** para la clave del servidor.
- **server.crt:** para el certificado.
- **ca.crt:** para la autoridad certificadora.

En otro caso la aplicación no podrá encontrar los certificados de seguridad.

Una vez hecho esto, procederemos a descargar e instalar los módulos de Node utilizados por la aplicación. Estos módulos están definidos en el fichero *package.json*. Para realizar esta operación, será necesario ejecutar el siguiente comando, utilizando la consola en el directorio raíz del proyecto:

>> n**pm install**

Después de ejecutar este comando aparecerá una nueva carpeta en el directorio con el nombre node_modules. En ella se encuentran los módulos requeridos.

Despliegue:

Para desplegar la aplicación basta con ejecutar el comando:

>> **node app.js**

Para comprobar que todo ha ido correctamente basta con acceder desde un navegador a la url:

https://[[dirección IP de la máquina]]:8001